REPORT DOCUMENTATION PAGE

AFRL-SR-BL-TR-98-

0196

88

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	06 Feb 1998	Final (01 Sep 94 – 31 Aug 97)

4. TITLE AND SUBTITLE	5. FUNDING NUMBERS
Robust Control Theory and Applications	F49620-94-1-0420
6. AUTHORS	
Professor John C. Doyle	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)	8. PERFORMING ORGANIZATION REPORT NUMBER
California Institute of Technology Pasadena, CA 91125	
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSORING/MONITORING AGENCY REPORT NUMBER
AFOSR/NM 110 Duncan Avenue, Room B-115 Bolling Air Force Base, DC 20332-8080	

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION AVAILABILITY STATEMENT	12b. DISTRIBUTION CODE
Approved for Public Release	

13. ABSTRACT *(Maximum 200 words)*

This AASERT program supported research in robust simulation, hierarchical uncertainty representation, and novel methods for robustness analysis of uncertain systems.

In the context of this program, robust simulation means simulating simultaneously sets of initial conditions and disturbance or noise signals. Thus sets of state space must be propagated by the dynamics of the model. Initial investigations have focused on piecewise linear discrete time systems, which map polyhedra to polyhedra at each time step. Linear programming can be used to refine the resulting bounds. This is important if the potentially exponential growth in set descriptions is to be overcome.

Hierarchical uncertainty modeling is a new framework to include explicit representation of uncertainty in component modeling. The focus has been on LFTs and implicit (DAE) representations. A variety of examples including parasitics and nonlinearities illustrate the key ideas.

Finally, this report describes new bounds on a "spherical μ" problem that allows for correlations between uncertainties in an LFT framework. Interestingly, this setting provides quite elegant bounds and simplified computation.

19980223 152

14. SUBJECT TERMS
robust simulation, hierarchical uncertainty representation

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Robust Control Theory and Applications

John C. Doyle
Division of Engineering and Applied Science
California Institute Institute of Technology

AFOSR Grant F49620-94-1-0420
Dr. Marc Jacobs, Program Manager

Final Report
6 February 1998

Summary of Completed Project

This AASERT program supported research in robust simulation, hierarchical uncertainty representation, and novel methods for robustness analysis of uncertain systems.

In the context of this program, robust simulation means simulating simultaneously sets of initial conditions and disturbance or noise signals. Thus sets of state space must be propogated by the dynamics of the model. Initial investigations have focused on piecewise linear discrete time systems, which map polyhedra to polyhedra at each time step. Linear programming can be used to approximation complicated polyhedra with simpler bounds, and branch and bound can be used to refine the resulting bounds. This is important if the potentially exponential growth in set descriptions is to be overcome.

Hierarchical uncertainty modeling is a new framework to include explicit representation of uncertainty in component modeling. The focus has been on LFTs and implicit (DAE) representations. A variety of examples including parasitics and nonlinearities illustrate the key ideas.

Finally, this report describes new bounds on a "spherical μ" problem that allows for correlations between uncertainties in an LFT framework. Interestingly, this setting provides quite elegant bounds and simplified computation.

1 Robust Simulation

1.1 Introduction

Historically, simulation has played a large role in nonlinear system analysis. Stability and performance are often studied by simulating a large number of initial conditions and noise signals. However, the results of these simulations do not guarantee stability or performance, since the behavior for other initial conditions and noise signals is not examined.

Robust simulation addresses this limitation. Instead of simulating a single initial condition and noise signal, a set initial conditions and noise signals are simulated at once. If the robust simulation meets the performance requirement, then the system meets the performance requirement.

As with any general nonlinear problem, some assumptions are required to ensure reasonable computational cost. For robust simulation, finite time horizon problems for discrete time piecewise linear systems are considered. All simulation based techniques require finite time horizons and the discrete time piecewise linear restriction is less onerous than it sounds.

Traditional simulation maps a single point in initial condition and noise space into a single final condition. We define robust simulation as the mapping of all initial conditions and noise signals into a set of final conditions. By calculating all possible trajectories, one can make guarantees about the system's global behavior. Traditional simulation only gives local information. Essentially, robust simulation maps sets to sets, while traditional simulation maps points to points.

The mapping of sets introduces a new problem, set representation, into the simulation process. For linear systems, this is not a difficult issue, since polyhedra map to polyhedra via matrix multiplication. However, nonlinear systems can map polyhedra into anything. For efficient computation, simple set descriptions and mapping rules are required.

The set representation issue is simplified by requiring discrete time piecewise linear systems, which map polyhedra to polyhedra via simple matrix operations. This simplification also eliminates two problems encountered in traditional simulation: step size determination and derivative approximation. This allows us to focus on the issues central to robust simulation, set representation and computational complexity.

In order to simulate sets, assumptions must be made on the form of system. By considering only PL systems, all mapping operations can be accomplished through linear programming and simple matrix operations. Since large linear programs are easily solved by existing software and linear program size grows reasonably with state dimension, large systems can be reliably simulated.

The idea of the algorithm is quite simple. Start with an initial condition set and noise description and calculate all possible final conditions after one time step. This is repeated until the desired final time is reached. While this simple approach calculates the exact set that can be reached at the final time, it also has exponential computational cost. Assuming that the initial set, $\mathcal{S}_0$, is a single polyhedra, the reachable set at any time t, denoted $\mathcal{S}_t$, may contain l^t polyhedra. Since $\mathcal{S}_0$ may intersect with each of the R_i, different subsets of $\mathcal{S}_0$ may follow up to l different state update laws, each yielding a separate polyhedra. Thus, $\mathcal{S}_1$ may

contain l polyhedra. Each polyhedra in $\mathcal{S}_1$ may map into l additional polyhedra, yielding up to l^2 polyhedra in $\mathcal{S}_2$. For long simulations, this exponential growth is unacceptable.

By restricting the number of polyhedra in each set $\mathcal{S}_k$, exponential growth is avoided. This is accomplished by combining polyhedra into larger polyhedra such that each of the original polyhedra is a subset of the new one. Thus, $\mathcal{S}_k$ contains all points reachable from $\mathcal{S}_{k-1}$ plus additional points added when combining polyhedra. This approximation is conservative. The simulation result always contains all reachable points, but may contain many points that are nor reachable. However, the conservatism can be systematically reduced by allowing more polyhedra in $\mathcal{S}_k$. When $\mathcal{S}_k$ can contain l^k polyhedra, the exact answer is obtained.

With the above approximation, robust simulation's computational cost grows linearly with time. At each time step, the same number of polyhedra are being mapped. Doubling the simulation length only doubles the computation time. Computational cost grows polynomially in other variables, as well. The approximation requires solving roughly $tnl^{\gamma+2}$ linear programs, where $0 \leq \gamma < t$ is the degree of refinement. For basic robust simulation ($\gamma = 0$) using a public domain $\mathcal{O}(n^4)$ linear program solver, the overall complexity is $\mathcal{O}(tl^2n^5)$. Memory usage is $\mathcal{O}(l^{\gamma+2}nd)$. These are worst case complexities; performance is generally much better than $\mathcal{O}(tl^2n^5)$.

1.2 Piecewise Linear Systems

When developing numerical tools for nonlinear analysis, piecewise linear (PL) systems are a natural class of systems to consider. They are a conceptually simple extension of linear systems. In any region of state space, an affine state update law is followed. PL systems are well suited to digital implementation and simulation; only matrix manipulations and if-then blocks are needed. PL systems can be completely described by a finite amount of information. Even the most complex PL system is described by a list of matrices.

As simple as they sound, PL systems are a very rich class. Many nonlinearities, such as saturations, dead zones, quantizations, and hysteresis, are naturally written as PL mappings. PL systems can exhibit very complex dynamics, including chaos. In fact, PL systems can approximate general nonlinear systems to any degree of accuracy.

Unfortunately, the simple intuition and ease of implementation does not extend to analysis. Calculating almost any interesting PL system property is NP-hard.[35] Decades of study have yielded few practical analysis results. In 1981, Sontag suggested the use of PL systems for nonlinear regulation [33] and developed a PL algebra [34]. Recently, Pettit and others have studied continuous time PL systems.[31] Unfortunately, these results have not yielded practical analysis tools. Computations either grow exponentially with problem size or do not give guarantees.

A piecewise linear (PL) system is defined over some subset, Z, of a finite dimensional real vector space, $\mathcal{R}^n$. Z is the union of a finite number, l, of closed polyhedra, denoted R_i. Each R_i is defined by a finite number of linear inequalities, $f(x) \geq a$. In each R_i, an affine state transition map is defined by

$$x(k+1) = A_i x(k) + B_i + N_i w[k], \quad x \in R_i, \quad i \in 1 \ldots l. \tag{1}$$

The noise signal, $w[k]$, is chosen to be in l_∞ to simplify computations. The methods presented trivially extend to time-varying PL systems, as well.

This definition simplifies computer implementation, but leads to a minor technical problem. Since the polyhedra are closed and may share boundaries, the mapping is not always well defined on the boundaries. While it is possible to create a PL system that exploits the behavior on the boundaries, this is generally not the case. Because this technical issue is easily resolved (see [33]), and only clutters the description of the problem, it will be ignored for the remainder of this presentation.

By construction, each R_i is convex and bounded by hyperplanes. While the number of hyperplanes, c, bounding each R_i may vary, c must be greater than $n+1$ (to define a simplex) and generally $c = 2n$ (to define a hyperrectangle). The notation $\mathcal{S}_j$ denotes a finite set of polyhedra in $\mathcal{R}^n$ at time j.

Linear programming and convex set theory play a large role in PL system analysis. By construction, each volume R_i is convex and bounded by hyperplanes. While the number of hyperplanes, c, bounding each region R_i may vary, c must be greater than $n+1$ (to define a simplex) and generally $c = 2n$ (to define a hyperrectangle). The hyperplanes bounding R_i, called $H_{i,h}$, $h \in 1 \ldots c$, are defined by a point, $x_{i,h}$, and a normal vector, $N_{i,h}$. The normal vector is defined as pointing into the region R_i. With this definition, linear programming can be used to solve a variety of problems. For example, a point x lies inside R_i if and only if

$$\langle x - x_{i,h} \cdot N_{i,h} \rangle > 0, \forall h \in 1 \ldots c$$

This can be written as a linear program with c constraints. Other problems, such as finding the intersection of regions and determining norms on regions are also linear programs.

1.3 Robust Simulation Algorithm

The robust simulation algorithm answers the following:

> For a given set of initial conditions, denoted $\mathcal{S}_0$, what are all possible final conditions, denoted $\mathcal{S}_f$?

By assigning a norm on $\mathcal{S}_f$, this becomes a performance measure.

The direct approach to calculating $\mathcal{S}_f$ has exponential growth in the computation as a function of t. To see this, start with $\mathcal{S}_0$, map forward one time step, and call the new set $\mathcal{S}_1$. Assuming (for notational simplicity) that $\mathcal{S}_0$ contains at most l convex regions, there are at most l^2 convex sets in the $\mathcal{S}_0 \cap R_i$. Since no restrictions are placed on (1), each of these convex sets can then map into all of the R_i. Thus, $\mathcal{S}_1$ can contain up to l^2 convex sets and $\mathcal{S}_1 \cap R_i$ may contain l^3 sets. At any time step j, $\mathcal{S}_j$ can contain as many as l^{j+1} independent convex sets. Repeating this process to form $\mathcal{S}_t$ yields, possibly, l^{t+1} sets. All of these mappings can be computed by simple matrix operations and linear programming.

By slightly modifying the direct approach, a polynomial time bound is obtained. The fundamental idea is to limit the number of regions in $\mathcal{S}_j$ at every time step by restricting

$\mathcal{S}_j$ to have a fixed number of convex sets in each R_i. The restrictions placed on $\mathcal{S}_j$ determine the tightness of the bound. The result contains $\mathcal{S}_f$, though the approximation may be conservative.

Given $\mathcal{S}_j$, the first step is to form a manageable approximation $\mathcal{T}_j \supseteq \mathcal{S}_j$. $\mathcal{S}_{j+1}$ is then calculated from $\mathcal{T}_j$. By restricting the number of sets in the approximation, exponential growth is avoided. By definition, $\mathcal{T}_j$ contains $l^{\gamma+1}$ convex sets, and there are l^{γ} convex sets in $\mathcal{T}_j \cap R_i$. The meaning of γ will be described later. Though the details are omitted due to space constraints, $\mathcal{T}_j$ can be calculated by linear programming.[12]

1.3.1 Algorithm Refinements

The accuracy of the approximation is directly related to the amount of extra volume added when forming $\mathcal{T}_j$. Two factors affect this: the number of convex sets in each R_i, and the number of hyperplanes, c, used to bound each region. By increasing each of these, accuracy may be improved.

As defined earlier, $\mathcal{T}_j \cap R_i$ contains l^{γ} convex sets. The value γ can be considered a history parameter. For $\gamma = 0$, one does not consider what regions a set mapped from before arriving in the current R_i. For $\gamma = 1$, one looks at where the set was one time step prior to the current time step. In this case, each $\mathcal{T}_j \cap R_i$ contains l convex sets, each approximating the sets that came from a specified R. γ determines how many previous time steps play a role in forming the approximation $\mathcal{T}_j$. As γ approaches t, the approximation approaches the exact solution. When $\gamma = t$, the exact solution is obtained.

An equally important variable is the number of hyperplanes defining a convex set, c. Assuming hyperrectangles, the basic algorithms required $2n$ bounding surfaces. In general, this is $2n(\gamma+1)$. The bounding hyperplanes must contain those of the current region, R_i, and the bounds of the previous regions considered after the mapping law is applied. This way, a hyperrectangle can be exactly covered after mapping γ time steps.

Since $c \geq 2n(\gamma + 1)$, an *ad hoc* approach for improving the approximation is to add more bounding surfaces according to some heuristic. In general, additional hyperplanes should differ greatly from existing hyperplanes.

Additionally, other generic bound improvement techniques, such as branch and bound are applicable.

1.3.2 Computational Complexity

The robust simulation approximation requires solving $tcl^{\gamma+2}$ linear programs. $\mathcal{O}(n^4)$ public domain linear program solvers are available and $\mathcal{O}(n^{3.5})$ algorithms have been proposed.[17] With $c = 2n$, the default when using hyperrectangles, $\gamma = 0$, and an $\mathcal{O}(n^4)$ linear program solver, the overall complexity is $\mathcal{O}(tl^2n^5)$.

Memory usage is $\mathcal{O}(l^{\gamma+2}nc)$. At any time, up to $l^{\gamma+2}$ linear programs must be stored in memory. These linear programs require memory proportional to nc. Since the results from previous time steps do need to be saved, memory usage is independent of t.

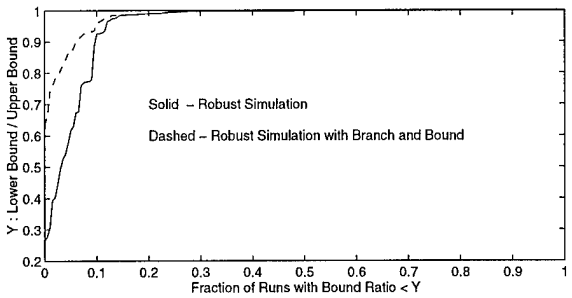

Figure 1: Nonlinear performance bound ratios

In general, performance is much better than $\mathcal{O}(tl^2n^5)$. This worst case performance assumes that each region R_i maps into every other region at each time step. Many systems map only into adjacent regions or themselves at each time step. By calculating what regions flow into other regions in advance, the number of linear programs solved at each step can be greatly reduced.

1.3.3 Examples

To evaluate the algorithm, tests were run on a set of randomly generated systems. The random systems were discretizations of 5th order continuous systems with one saturation nonlinearity and no noise. Systems were simulated for 30 time steps. The performance measure chosen was $|x[30]|_\infty$. The initial condition set was $|x[0]|_\infty \leq 1.3$. This is a reasonable set of test problems that are moderately challenging for the algorithm. This is neither the hardest nor the easiest class of problems known.

Any traditional simulation gives a lower bound on the worst case performance. To achieve a large lower bound, gradient search was combined with random simulation. The upper bound was calculated by robust simulation with $\gamma = 0$. If the bounds differed by more than 10%, naive branch and bound was applied until the bounds differed by less than 10% or 50 branch steps were taken.

The measure of algorithm performance, shown in Figure 1, is the ratio of the lower bound to the upper bound. Ideally, the ratio is always 1. In general, ratios greater than 0.9 are acceptable. For 89% of the runs, no branching was needed to obtain acceptable results. For the 11% of the runs where branching was needed, branch and bound greatly improved the bound ratio. These results can be greatly improved by better selection of branch cuts.

1.4 Robust Stability of PL Systems

Efficient algorithms for verifying the stability of uncertain discrete time piecewise linear (PL) systems will be presented. While PL systems are intuitively simple, they are computationally hard. Two approaches to verifying stability are presented. For each approach, separate necessary and sufficient conditions are given.

The first approach requires the solution of a linear matrix inequality. This method is only applicable to a restricted class of PL systems, and is generally very conservative. It is demonstrated that for most PL systems, these conditions yield no information.

The second, more general, approach is based upon robust simulation. This method is useful for all PL systems, and will always yield a definitive answer. If a system initially satisfies necessity, but fails sufficiency, these algorithms can be systematically refined and after a finite number of refinements, a definitive answer is guaranteed.

The algorithms are illustrated on four examples, including examples where no other general analysis technique is available.

1.4.1 LMI based Alogorithm

For a restricted class of PL systems, separate necessary and sufficient conditions for stability require solving LMI's. However, these conditions are very conservative.

For PL systems without any affine terms ($B_i = 0 \ \forall i$), stability can be guaranteed by finding a feasible solution, V, to the LMI

$$\begin{aligned} V &> 0 \\ A_i^t V A_i - V &< 0, \ i \in 1 \ldots l. \end{aligned} \tag{2}$$

Similarly, instability is guaranteed if a feasible solution to

$$\begin{aligned} V &> 0 \\ A_i^t V A_i - V &> 0, \ i \in 1 \ldots l \end{aligned} \tag{3}$$

exists.

If (2) has a solution, then $x[k]^t V x[k]$ is a common quadratic Lyapunov function for all A_i and the PL system is stable. This Lyapunov function, however, is independent of the switching law. Switches can occur arbitrarily quickly or even randomly, and the PL system is still stable. Clearly, this is sufficient condition for stability, though it is not necessary. In section 1.4.3, an example is given that fails this test, but is still stable.

For stability, it is also necessary that there is no solution to (3). If a solution exists, then $\mathcal{V}(k) = x[k]^t V x[k]$ is Lyapunov-like function where $\mathcal{V}(k+1) > \mathcal{V}(k) \ \forall k$, and the system is unstable for any switching law.

As previously mentioned, these conditions are very conservative. If any A_i has an unstable pole, then (2) never has a feasible solution. One can easily construct a stable PL system where some A_i are unstable. If any A_i is stable, then (3) never has a solution. Section 1.4.3 presents a PL system where all A_i are stable, but the system is unstable. These results are closely related to discrete linear inclusions and are well known. [3, 10]

1.4.2 Robust Simulation Algorithm

The robust simulation based algorithm examines generalized finite time horizon stability, specifically:

Do all states in a set of convex regions $\mathcal{S}_0$ map into a set of convex regions $\mathcal{S}_f$ within t time time steps?

With a careful choice of $\mathcal{S}_f$, stronger notions of stability can be inferred. If $\mathcal{S}_f$ is known to be invariant, then Lyapunov stability is implied. If $\mathcal{S}_f$ is known to be invariant and a quadratic Lyapunov function exists for $\mathcal{S}_f$, then quadratic stability is implied. The latter case occurs frequently, often when $0 \subset \mathcal{S}_f \subset R_i$ for a fixed i. Answering this question is a direct application of robust simulation, which calculates all possible trajectories for a set of initial conditions.

A sufficient condition for generalized stability is obtained by robustly simulating $\mathcal{S}_0$ for t time steps, yielding the set of all possible final points, $\mathcal{S}_t$. If $\mathcal{S}_t \subseteq \mathcal{S}_f$, then the stability condition is satisfied. It should be noted that robust simulation is conservative; $\mathcal{S}_t$ contains all possible final conditions and additional points that cannot be reached from $\mathcal{S}_0$. This condition only guarantees stability; if it is not met, nothing can be inferred.

A necessary condition for generalized stability is found by robustly simulating $\mathcal{S}_f$ for t time steps backwards in time, yielding the set $\mathcal{S}_{f-t}$. Robust simulation in backwards time is always possible, even if some A_i are not invertible. If $\mathcal{S}_0 \not\subseteq \mathcal{S}_{f-t}$, then the system not stable. When $\mathcal{S}_0 \not\subseteq \mathcal{S}_{f-t}$, some point in $\mathcal{S}_0$ does not map into $\mathcal{S}_f$ after t time steps. Since $\mathcal{S}_{f-t}$ contains points that do not map into $\mathcal{S}_f$, this is only a necessary condition.

If the sufficient condition fails and the necessary condition is satisfied, no conclusion can immediately be drawn. However, the conditions can be refined by using more accurate robust simulation algorithms at the expense of computational cost. The exact answer can always be obtained in finite time.

The generalized stability problem can easily be rephrased as an $\exists$ problem by checking for instability. For PL systems, $\exists$ problems are in general NP-hard.[35] Thus, an efficient algorithm that gives the exact answer is not expected. The question can be shown false in polynomial time by choosing an appropriate x_0 and calculating is trajectory.

These algorithms extend directly to uncertain (noisy) PL systems. Robust simulation can be used to analyze systems with flows of the form

$$x(k+1) = A_i x(k) + B_i + U_i \delta(k), \tag{4}$$

with $U_i \in \mathcal{R}^{nxq}$ and $\delta \in l_\infty$. Both the necessary condition and the sufficient condition derived in this section hold for this type of uncertain system. Robust simulation for noisy systems calculates all possible trajectories for all possible noise signals. The LMI based stability conditions are not applicable for this class of systems.

1.4.3 Examples

Two simple 2-dimensional examples show the limitations of the LMI approach and the value of the robust simulation based algorithms. Consider two piecewise linear systems, each with two regions ($l = 2$) whose flows are given by

$$A_0 = \begin{bmatrix} 0.81 & 0.22 \\ -0.22 & 1.17 \end{bmatrix}; \; A_1 = \begin{bmatrix} 0.81 & -0.22 \\ 0.22 & 1.17 \end{bmatrix}$$

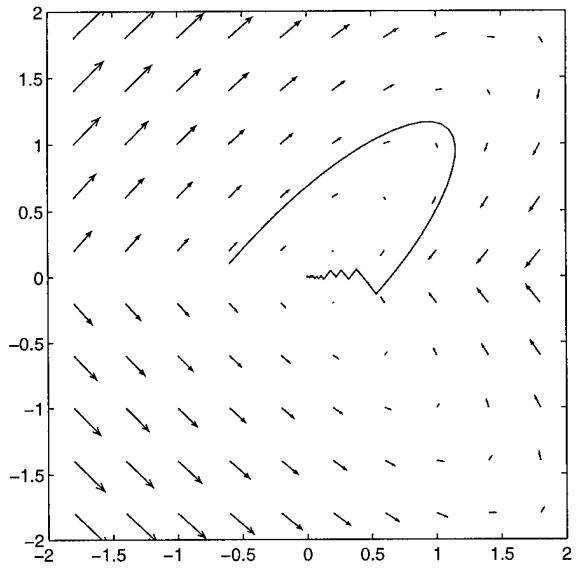

Figure 2: Stable PL system

with $B_0 = B_1 = [0; 0]$. The piecewise linear systems are defined on the rectangle $|x_1| \leq 100$ and $|x_2| \leq 100$. For both systems, the initial region, $\mathcal{S}_0$, is the rectangle defined by $|x_1| \leq 10$ and $|x_2| \leq 10$ and the final region, $\mathcal{S}_t$, is the rectangle defined by $x, x_1 \leq 0.1$ and $x_2 \leq 0.1$. It is important that the initial region, $\mathcal{S}_0$, is not the entire region of definition of the PL system. In both of these examples, some trajectories leave $\mathcal{S}_0$ and later reenter it. If the trajectory exited the system's region of definition, it would be unstable, by definition. The number of time steps, t, for both simulations is 100.

Since $B_0 = B_1 = [0; 0]$, the LMI approach is applicable to any piecewise linear system with these flows. However, both (2) and (3) have no feasible solution and no conclusion can be drawn. Though both A_0 and A_1 are separately stable, their combination may not be.

The first system follows the switching law

$$x[k+1] = \begin{cases} A_0 x[k] & x_2 \geq 0 \\ A_1 x[k] & x_2 < 0 \end{cases}$$

A sample trajectory, shown in figure 2, converges to the origin, staying close to the switching surface. The system satisfies the robust simulation based sufficient condition and is thus verified to be stable. The necessary condition is also satisfied, as expected.

The second system follows the switching law

$$x[k+1] = \begin{cases} A_0 x[k] & x_1 \geq 0 \\ A_1 x[k] & x_1 < 0 \end{cases}$$

A sample trajectory, shown in figure 3, diverges, oscillating along the switching surface. The system fails the robust simulation based necessary condition, and is thus verified to be unstable. The systems also fails the sufficient condition.

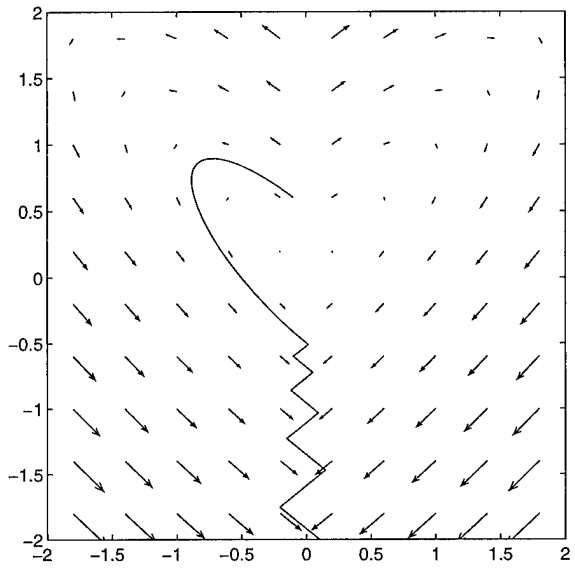

Figure 3: Unstable PL system

In both of these examples, the switching law was critical to the stability of the system. In any system where the switching law matters, the LMI stability conditions will never give any information since it explicitly ignores the switches. Though 2-dimensional systems can be analyzed by a variety of other methods, those methods do not generalize to higher dimension. Both the robust simulation based algorithms and the LMI approach generalize to arbitrary dimension.

A third 2-dimensional example demonstrates how bounded noise can drive a piecewise linear system unstable. Consider the piecewise linear system

$$x[k+1] = \begin{cases} A_0 x[k] & |x_1| \leq 0.1 \\ A_1 x[k] + B_1 + U_1 \delta[k] & x_1 < -0.1 \\ A_2 x[k] + B_2 & x_1 > 0.1 \end{cases}$$

where

$$A_0 = \begin{bmatrix} 0.9 & 0 \\ 0 & 0.9 \end{bmatrix}, \; A_1 = A_2 = \begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix},$$

$$B_1 = \begin{bmatrix} 0.1 \\ 0.1 \end{bmatrix}, \; B_2 = \begin{bmatrix} -0.1 \\ 0.1 \end{bmatrix}.$$

The piecewise linear system is defined on the rectangle $|x_1| \leq 100$ and $|x_2| \leq 100$. For the stability question, the initial region, $\mathcal{S}_0$, is the rectangle defined by $|x_1| \leq 10$ and $|x_2| \leq 10$. and the final region, $\mathcal{S}_t$, is the rectangle defined by $|x_1| \leq 0.1$ and $|x_2| \leq 0.1$. The number of time steps, t, is 100. Since some flows have affine (B_i) terms, the LMI based approach is not applicable.

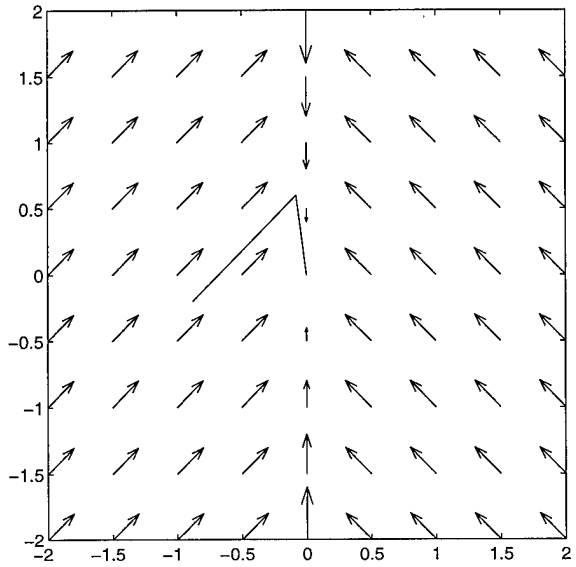

Figure 4: PL system trajectory without noise

With no noise ($U_1 = [0; 0]$), the system satisfies the robust simulation based sufficient condition and is stable. As shown in figure 4, flows starting with $|x_1| > 0.1$ move towards $x_1 = 0$, and then converge to the origin. In fact, once $|x_1| \leq 0.1$, the state converges to the origin quadratically.

When bounded noise is added as $U_1 = [0.1; 0]$, the system is destabilized. By choosing an initial condition in region 1 and a noise signal $\delta[k] = -1$, the system diverges, as shown in figure 5. This system fails both the sufficient condition and the necessary condition, and we conclude that it is unstable.

The next example demonstrated difference between nominal stability and robust stability for PL systems. Though all initial conditions converge to the origin in the absence of noise, a bounded noise can destabilize the system. No other techniques exist for analyzing PL systems with noise. Robust simulation accounts for all possible noise signals and provides the answer in reasonable time.

To demonstrate the algorithms on larger systems, a 5th order continuous linear system with an input saturation was randomly generated and discretized. Since the system has a saturation, some regions have affine terms and the LPV methods are not applicable. Rather than examine stability directly, the accuracy of robust simulation was examined.

The simulation started with the hypercube $|x[0]|_\infty \leq 1.3$ and ran for 20 time steps. At each time step, the difference between robust simulation and the exact answer was measured. The relative error, defined as $(|x_{simulated}|_\infty - |x_{exact}|_\infty)/|x_{exact}|_\infty)$, is shown in figure 6.

For this example, the difference between the exact solution and robust simulation is negligible. While there are systems for which basic robust simulation differs greatly from the exact answer, most discretizations of continuous systems give good results. While this is a

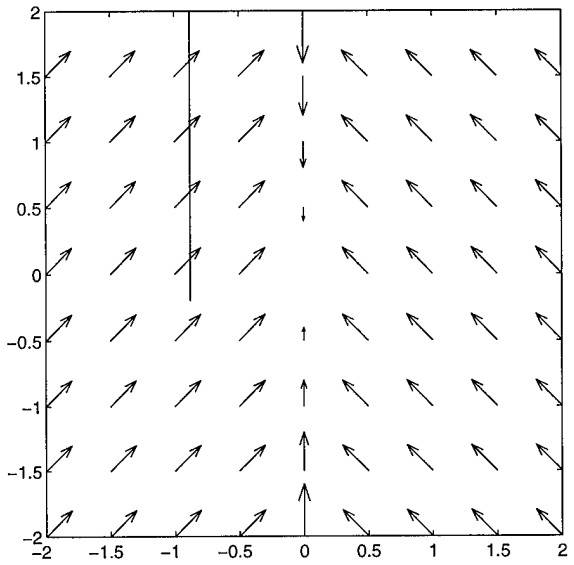

Figure 5: PL system trajectory with noise

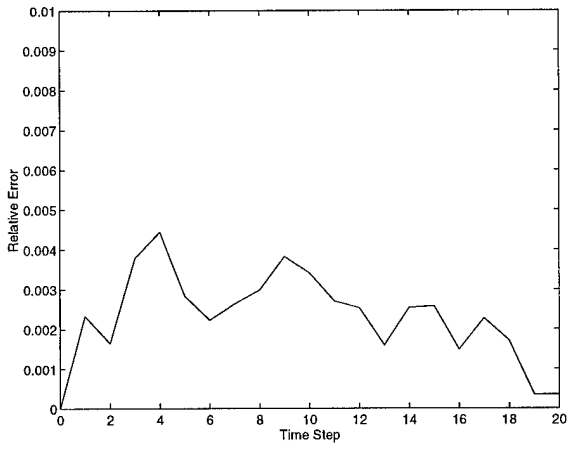

Figure 6: Relative error of robust simulation

relatively small system, it gives hope that the algorithms will perform well on large systems.

1.5 Nonlinear MPC Lower Bound

Model predictive control (MPC) for nonlinear systems typically involves a non-convex optimization problem. As with all non-convex optimizations, a local minimum is found, but nothing can be said about the global minimum. With a careful choice of cost, constraints, and system representation, robust simulation gives a lower bound on the optimal cost. If this bound differs greatly from the MPC cost, then additional optimization may be desired. Furthermore, the robust simulation results can be used to initialize additional MPC optimizations. This technique is demonstrated on a simple example.

1.5.1 Model Predictive Control

Model predictive control (MPC), also known as receding horizon control, is a technique where an on-line, open-loop control problem is solved at each time step.[8] Using the current state, an input sequence is calculated to minimize a cost while satisfying specified constraints. Only the first element of the sequence is used, and then the algorithm is applied again, beginning at the new current state. For general nonlinear systems, this technique results in a constrained non-convex optimization.[21] At each time step, a local minimum of the cost is found, but no information about the global minimum is provided.

By restricting the class of systems considered, iterative robust simulation can be used to find a lower bound on the global minimum. If the MPC cost and the robust simulation bound are similar, then additional optimization will yield little benefit. If they differ greatly, then additional optimization may yield large improvements. Robust simulation results are used to generate initial conditions for additional optimization runs, which will generally converge to a better local minimum.

To allow for analysis by robust simulation, we require that the system is piecewise linear (PL). PL systems, described in more detail in [15] and [34], behave as different affine systems in various regions of state space. In any region i of state space, the state transition map is defined by

$$x[k+1] = A_i x[k] + \bar{A}_i + B_i u[k]. \tag{5}$$

Secondly, we require the input constraint $|u[k]| \leq 1 \ \forall k$. Finally, the cost, J, must be a weighted infinity norm on the state error trajectory. Specifically,

$$J = \max_{k \in k_0 \ldots k_f} |W[k](x[k] - x_0[k])|_\infty. \tag{6}$$

Through $W[k]$, we can penalize later errors more than the initial transient response and penalize certain directions in state space more heavily than others. Level sets of this cost define a tube through which the state trajectory must pass.

1.5.2 Algorithm

The idea of the algorithm is to find the largest cost J, denoted J_{opt}, such that there are no feasible trajectories with cost less than J. This is accomplished by iterative robust simulation. Robust simulation, defined in [15] and described in more detail in [12], allows the calculation of all feasible trajectories.

The algorithm uses the following notation:

$$0 \leq J_{lb} \leq J_{opt} \leq J_{mpc}.$$

J_{mpc} is the cost from the MPC optimization and represents the lowest cost from a single, known feasible trajectory. J_{opt} is the unknown optimal cost. J_{lb} is a lower bound on J_{opt}. The main goal of the algorithm is to find a large J_{lb} with a secondary goal being to reduce the gap between J_{lb} and J_{mpc}.

The first step is to construct an MPC control sequence and calculate its cost, J_{mpc}. Since this is a feasible trajectory, it provides an upper bound on J_{opt}. Initially, $J_{lb} = 0$, since 0 is always a lower bound.

The second step is to choose a cost $J_{nom}, J_{lb} \leq J_{nom} \leq J_{mpc}$, and determine if any feasible trajectories meeting that cost exist. This is accomplished using a modified robust simulation algorithm that calculates the set of states that can be reached at each time step while satisfying the cost constraint.

Robust simulation is a discrete simulation technique that calculates all reachable states at time k_{i+1} from a set of states defined at time k_i. When finding MPC lower bounds, we are only concerned with states that are reachable and meet the cost constraint. With this added requirement, robust simulation is applied to the intersection of the set of states reachable at time k_i and the set of states meeting the cost constraint. Forming this intersection is equivalent to finding the intersection of two polyhedra and can be written as a linear program. When this intersection is empty, there are no trajectories satisfying $J \leq J_{nom}$ and the robust simulation algorithm is terminated.

If no feasible trajectory satisfying $J \leq J_{nom}$ exists, then J_{nom} is a lower bound on J_{opt}. If J_{nom} is an acceptable lower bound ($J_{nom} \approx J_{mpc}$) then we are done. If J_{nom} is too small, then set $J_{lb} = J_{nom}$ and repeat the second step.

If robust simulation generates potential trajectories that meet the cost constraint, then two options exist. Either the results from the robust simulation can be used to find a better MPC sequence (reduce J_{mpc}) or the robust simulation step can be repeated with a smaller J_{nom}.

The algorithm terminates when one is satisfied with J_{lb}. Typically, this is when $J_{mpc} - J_{lb}$ is small. Several decisions in the algorithm, including the choice of J_{nom}, are heuristic based. The use of robust simulation results to initialize added optimizations involves using branch and bound on the input space.

Since robust simulation's computational cost grows polynomially with problem size, this algorithm also exhibits polynomial growth. Other techniques for finding better local minima, such as gridding the parameter space, have exponential computational growth.

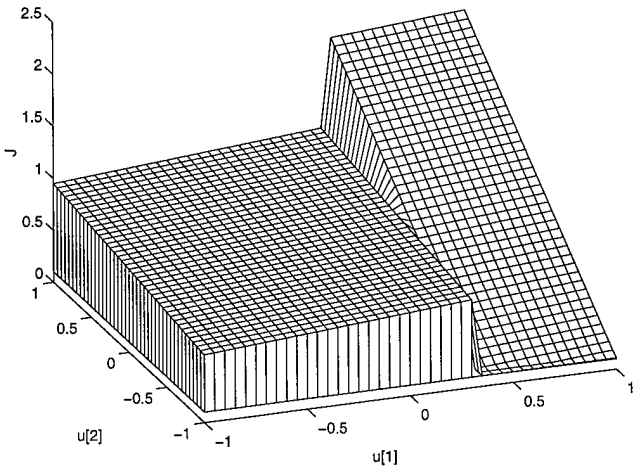

Figure 7: cost J as a function of $u[1]$ and $u[2]$

1.5.3 Example

Consider the piecewise linear system

$$x[k+1] = \begin{cases} x[k] + 0.15u[k], & x[k] < 1 \\ x[k] + u[k], & x[k] \geq 1 \end{cases}$$

with input constraints $|u[k]| \leq 1$. Starting from $x[0] = 0.95$, we want to minimize the cost $J = \max\{0.1x[1], x[2]\}$. The cost, J, as a function of the control inputs $u[1]$ and $u[2]$ is shown in Figure 7.

Any optimization scheme using a gradient descent algorithm and starting from $u[1] < 1/3$ will fall into the local minimum at $u[1] = u[2] = -1$. MPC based control schemes are vulnerable to failures of this sort in their optimization step [24].

The local minimum at $u[1] = u[2] = -1$ has a cost $J = 0.65$. This is more than a factor of 6 greater than the global minimum cost of 0.1, which occurs at $u[1] = 1/3, -1 \leq u[2] \leq -0.9$.

Starting the MPC optimization from $u[1] = u[2] = 0$ converges to the local minimum, resulting in $J_{mpc} = 0.65$. Iterative robust simulation gives a lower bound of $J_{lb} = 0.099$. The gap between J_{mpc} and J_{lb} indicates that the MPC solution may only be a local minimum. A single branch and bound step of the robust simulation suggests that $u[1] = 0.5, u[2] = 0$ is a better initial condition for the optimization. A gradient descent algorithm applied from this point quickly converges to a global minimum, giving a new cost $J_{mpc} = 0.1$. Since $J_{lb} \approx J_{mpc}$, further optimizations will yield little benefit.

While this is a simple optimization problem that is easily solved by gridding the parameter space, it demonstrates the feasibility of the algorithm. Future work will include applying this method to more realistic problems.

2 Hierarchical Modelling

For modeling complex systems, it is natural to reduce the system into subsystems and model each subsystem. The approach taken in this work is that it is desired for a model to be consistent with the modeling methodology. Further it is important to explicitly represent the inaccuracies of the model as part of the model.

A hierarchy, interconnection structure, and a fundamental component data type are proposed and the choices motivated. The framework is proposed with the intention of being implemented on a computer and having a family of models of different resolution representing a system.

There are many issues with this problem. One of the issues is variable granularity of the hierarchical model. What is meant by granularity is inherently scale. Do I have a simple aggregate model for a complex system, or do I need to model each of it's components in detail? With this sort of variable resolution, there are associated changes in the state dimension of the model of the system. These sorts of tradeoff in choosing a model should occur in a distributed manner.

A more specific and well defined issue, is the hierarchical modelling of static nonlinearities in a dynamical system. With a notion of the complexity for the approximation of a nonlinearity, there is a tradeoff between fidelity and complexity. One implementation of this solution is to model using sets of uncertain piecewise linear (PL) systems and simulate using robust simulation. Sets of uncertain PL models are simple to create and admit a simple partial ordering of their accuracy. Robust simulation explicitly accounts for model uncertainty and allows for switching between models during a single simulation. These techniques present a practical, computationally tractable solution to the problems encountered when simulating large systems.

2.1 Background

2.1.1 Linear Fractional Transformations

A linear fractional transformation is shown in Figure 8 for the map $y = (\Delta \star M)u$ where $\Delta \star M = D + C\Delta(I - A\Delta)^{-1}B$. In general, Δ represents uncertainty and dynamic elements,

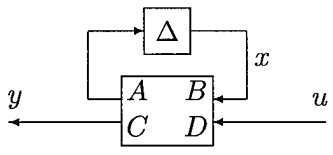

Figure 8: Linear Fractional Transformation

and $M = \begin{bmatrix} A & B \\ C & D \end{bmatrix}$ is a realization of the map $\Delta \star M$. The LFT framework results in a convenient method for adding various types of uncertainty to systems [42].

If $\Delta = \int$ operator then the system shown in Figure 8 is just the standard state space representation. LFT systems are a natural generalization of state space representations. By allowing Δ to represent more general systems operators, LFT systems provide a convenient framework to add various types of uncertainty operators, like Nonlinear, LTV, LTI, LPV, etc [28] [40] [6], in which essentially all the major state space results can be generalized [1].

2.1.2 Implicit LFT Systems

An implicit LFT system is described by $0 = (\Delta \star M)w$ as shown in Figure 9, where w contains all the system variables ie. there is no distinction between inputs and outputs. Implicit LFT systems are a generalization of the behavioral framework proposed by Willems [37].

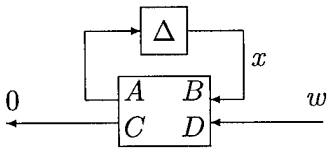

Figure 9: Implicit LFT system

Any LFT system can be converted into an implicit LFT system. In first principles modeling, like $F = Ma$, neither F or a is assumed to be an input or an output. Once either F or a is defined, both are defined. This is natural within the implicit LFT form but doesn't fit and makes interconnections difficult within the LFT form.

For tree structured hierarchical models, at the interconnections there isn't a notion of signal flow. The interconnection variables become internal variables to the system rather than inputs or outputs. So it is more natural to not make the distinction between inputs and outputs in modeling.

Within the implicit LFT framework, the interconnection of systems is simple. The implicit description of a system describes the equations a system must satisfy ($0 = (\Delta_i \star M_i)w_i$ where $M_i = \begin{bmatrix} A_i & B_i \\ C_i & D_i \end{bmatrix}$). So if two systems are connected then they still satisfy the same equations, and in addition the interconnection must be defined ($T_1 w_1 + T_2 w_2 = 0$) which defines the intersection of behaviors [37]. So the implicit LFT model of the interconnected system is given by $0 = (\Delta \star M)w$ where:

$$M = \begin{bmatrix} A_1 & 0 & B_1 & 0 \\ 0 & A_2 & 0 & B_2 \\ C_1 & 0 & D_1 & 0 \\ 0 & C_2 & 0 & D_2 \\ 0 & 0 & T_1 & T_2 \end{bmatrix} \tag{7}$$

$$\Delta = \begin{bmatrix} \Delta_1 & 0 \\ 0 & \Delta_2 \end{bmatrix} \tag{8}$$

Implicit representations can be interconnected, manipulated and reduced without committing to a particular input-output form, which is only relevant to certain applications, and which can be easily derived a posteriori if necessary as shown in D'Andrea and Paganini [5].

The implicit LFT framework also allows our model to include integral quadratic constraints (IQCs)[22]. IQCs are inequalities involving a quadratic form in signal space:

$$< \Pi w, w >= \int_{-\infty}^{\infty} w(\omega)^* \Pi w(\omega) d\omega \leq 0 \tag{9}$$

where Π^* is an LTI operator. IQCs can be used to define sets in which signals must exist, like defining sets of allowable noise. Given $\Pi(e^{j\omega}) = \Pi(e^{j\omega})^* \in L_\infty \Rightarrow \exists k > 0 \ni kI + \Pi > 0$. By doing a spectral factorization [42] of $kI + \Pi$, we find a $Q \ni \Pi = kI - Q^*Q$. Defining $P = k^{1/2}I \Rightarrow \Pi = P^*P - Q^*Q$. Equation 9 becomes $\| Qw \|_2 \geq \| Pw \|_2$. This can be written as an uncertain implicit equation of the form $(P + \Delta_c Q)w = 0$ where Δ_c is an arbitrary contractive operator [29].

2.2 General Paradigm

A modeling framework involves choices. This work presents and motivates the choices made in this uncertain hierarchical modeling framework. The motivation for choosing hierarchical modeling is that not all components of a system are equally significant and this should be reflected in the model. In the case of a car, the ashtray is a less significant component than the engine. If a hierarchical model is not used then all the system equations are written at the same level, and it may be difficult to immediately distinguish the dominant dynamics of the system from the trivial dynamics. The hierarchy hopefully provides a quick and efficient method for identifying the significance of dynamics and doing model reduction as desired.

The second choice made is doing uncertain modeling. For a real system, finding an exact model is impossible and characterizing the inexactness is critical for making guarantees on system performance. In the case of a resistor, there is uncertainty on the exact value of the resistance and the parasitics. When a model is reduced, the reduced dynamics must be covered with uncertainty. This may be useful when cruder models with uncertainty are sufficient for a particular application which may lead to reduced computation.

The next choice made is a tree structured hierarchy. The motivation for a tree structure is that a complicated system is naturally decomposed into an interconnection of simpler, more tractable subsystems and each subsystem can be similarly reduced. This self similarity leads to a tree structure. For example, modeling all the facets of a car is quite a task, but a more natural approach is to break up the car into more tractable components and model them individually. So rather than modeling the entire car, a car is an interconnection of an engine, transmission, exhaust system, cooling system, suspension, etc and each of the components is modeled separately.

A benefit of the tree structure is the connection with the object oriented philosophy. The tree defines how the components interact with each other. Which simplifies future modifications, because if a system is modified only the modified components needs to be remodeled, and the other component models will remain intact. So in the car example, if the engine is

replaced with a different model, the entire system doesn't have to be remodeled. The old engine model is replaced by the model of the new engine.

Another choice that is made is that the model will be constructed, implemented and used on a computer. As a result, the data structures should be convenient and tractable for computer implementation as opposed to writing them out by hand. The reason for this choice is that as more complicated systems have more and more detailed models and are to be analyzed or simulated it is intractable to do them any other way.

The proposed framework is defined by a hierarchical tree structure of the components, an interconnection structure of the components, and a fundamental data type for a component. Throughout this work, an inductor is used to demonstrate the features of the framework. An inductor is a simple example, but is necessary to make the presentation tractable.

2.2.1 Component Modeling

Consider the ideal inductor shown in Figure 10 with the equation $d/dt(Li) = v$ with in-

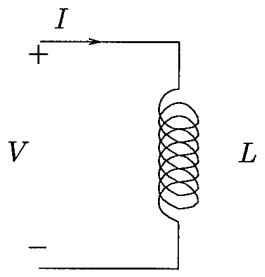

Figure 10: Inductor

ductance L, where v is an input, i is an output, and ϕ is the flux in the inductor. The resulting LFT model (Figure 8) of the system is given by $u = v$, $x = \phi$, $y = i$, $\Delta = \int$, and $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{c|c} 0 & 1 \\ \hline 1/L & 0 \end{array}\right]$ As a result of choosing this model of an inductor, two problems arise. First, if the effects of uncertainty in L, nonlinearities and parasitics, are to be investigated, there is no convenient way to do this without starting over. Second, by assuming that v is an input and i is an output, our model may be incompatible with other components with which it is interconnected as shown in Figure 11. Figure 11 represents an input/output interpretation of two inductors connected in series ($i_1 = i_2$). The connection of the two outputs is not properly defined within the input/output framework.

The solution to the first problem is easily addressed in the LFT framework. For the inductor example in Figure 10, the modeled equations should be replaced by $d\phi/dt = v$ and $\phi = L(I) = L_0(1 + \delta)i$. Where L_0 is a constant which represents the nominal value of the inductor and δ is an unknown operator. The model of the uncertain inductor is given by

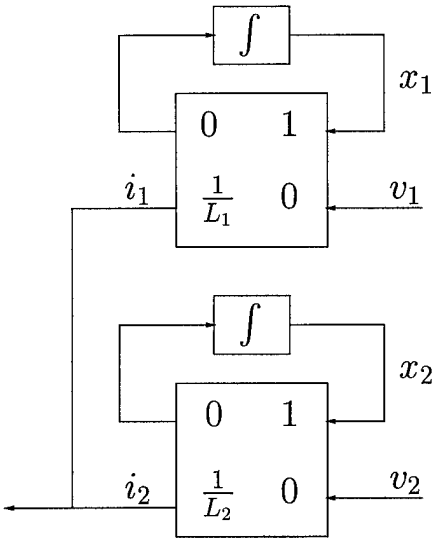

Figure 11: Series Interconnection of Inductors

$y = i$, $x = [\phi, x_2]$, $u = v$, $\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{cc|c} 0 & 0 & 1 \\ 1/L_0 & -1/L_0 & 0 \\ \hline 1/L_0 & -1/L_0 & 0 \end{array}\right]$, and $\Delta = \left[\begin{array}{cc} \int & 0 \\ 0 & \delta \end{array}\right]$. x_2 is not a state in the conventional sense but the output δ [1]. δ could represent time variation in the inductance due to saturation, variations in the core geometry, magnetic links to other components, nonlinearities, etc.

LFTs provide a flexible modeling framework for incorporating uncertainty descriptions, but the input-output assumption is not desirable for the modeling of interconnected systems. It is not a priori known which variables should be treated as inputs and which should be treated as outputs [37].

To address the second problem, systems will be represented in an implicit LFT form where $0 = (\Delta \star M)w$ as shown in Figure 9. There is no issue of compatibility between implicit LFT models of components because no input-output partition has been made.

For the inductor example, the implicit model of the inductor is given by $d\phi/dt = v$, $L_0(1+\delta)(i) = \phi$, $w = [v, i]$,

$$\left[\begin{array}{c|c} A & B \\ \hline C & D \end{array}\right] = \left[\begin{array}{cc|cc} 0 & L_0 & 0 & L_0 \\ 0 & 0 & 1 & 0 \\ \hline -1 & 0 & 1 & 0 \end{array}\right], \text{and } \Delta = \left[\begin{array}{cc} \frac{d}{dt} & 0 \\ 0 & \delta \end{array}\right].$$

For the fundamental components in this modeling system, the model must be general enough to describe any situation which may occur. The model should have some information about the component, so that each component isn't just an arbitrary operator. The information is contained in the nominal value of the component. No irreversible assumptions should

be made. For example, the standard circuit equation for an inductor is $V = L \cdot di/dt$. If this were the fundamental model of an inductor, the assumption has been made that L is a constant. If L were blindly replaced by $L(t)$ the resulting model would be incorrect. The actual equations for an inductor are given in equations 10 and 11.

$$\phi = Li \tag{10}$$

$$kv = d\phi/dt \tag{11}$$

Once we have our model and are ready to do analysis, synthesis, or simulation any assumptions can be made about our uncertainty like Δ_L is a real parameter, a bounded operator, etc, but this is after the modeling process and a part of the identification process.

We want to make the weakest possible assumptions about our inductor (L,k), so that wide variety of uncertainty assumptions can be made at the analysis level. L and k are assumed to be a non-commuting indeterminates (NCIs), ie. it could be an arbitrary nonlinear, time-varying operator. For a real inductor, it's nominal value is of use in describing it's operation. For modeling, $L := L_0 + \Delta_L$ and $k := 1 + \Delta_v$, where Δ_L and Δ_v are NCIs and L_0 is a "place holder" for the nominal inductance. L_0 is used to describe the ideal model of an inductor. NCI's act as "place holders" for uncertainty descriptions. It is important to note that by setting $L := L_0 + \Delta_L$ we have not committed to anything. This can be undone by defining $\Delta_L := -L_0 + \Delta_{Lnew}$. The L_0 term is added because the nominal value is presumably useful in describing the operation of the system.

Part of the modeling process is the addition on new components (like adding parasitics to the model of a inductor). It is desired that the current model can be refined to arrive at a new model rather than discarding the model and starting over. As a result the model must have "place holders" for defining new interconnections. If these "place holders" do not exist then it is possible that at an interconnection an incorrect equation may result as shown in Figure 12. Figure 5a would lead to the constraint that $i_1 = i_2$ but if later an additional component is added as in Figure 5b then $i_1 = i_2 + i_3$. These two constraints lead to $i_3 = 0$ which may be incorrect.

In the circuit case, these "place holders" would correspond to error currents for each node and error voltages for each branch connecting nodes. These are needed to describe all the possible interconnection of circuits. When we do the analysis these error place holders must be resolved. They can be set to 0, ie. no additional current into a node, they can be considered external noise terms and constrained according to IQCs, or used to cover unmodeled dynamics.

2.2.2 Component Data Type

A component is chosen to be modeled by an implicit LFT system (Figure 9) with special structure. The implicit LFT structure is chosen because it provides a powerful framework for describing uncertainty plus it is natural for interconnecting systems. The system variables w are partitioned as:

$$w = \begin{bmatrix} w_m & w_n & l_m & l_c & l_e \end{bmatrix} \tag{12}$$

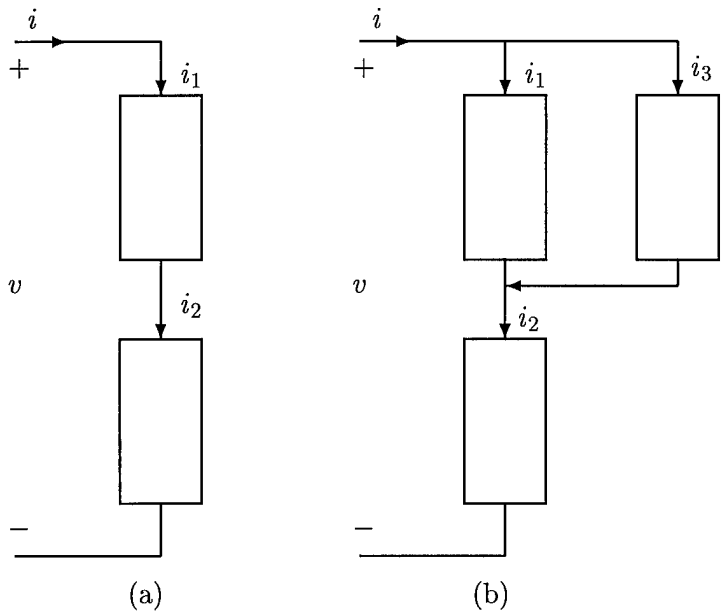

Figure 12: Incorrect and Irreversible interconnections

Where w_m are the manifest variables of the component. They represent the variables that are connected to the parent component. For an inductor, $w_m = \begin{bmatrix} v & i \end{bmatrix}$. w_n are the nominal manifest variables, and l_m are the latent manifest variables. w_n are used to describe the nominal dynamics of the component. w_n and l_m are used to describe the interconnection of the nominal component with the child components. l_c are the latent crosstalk variables, and are the terms used to describe crosstalk connections (connection between unrelated components). l_e are the latent error variables, they are used to describe interconnection errors and act as "place holders" for describing future interconnections. It is difficult to motivate this partition of the system variables, but this partition should be clear after the hierarchy and interconnection structure of the system is defined.

The special structure of $\Delta \star M$ is described by:

$$B = \begin{bmatrix} 0 & b_1 & 0 & b_2 & b_3 \end{bmatrix} \tag{13}$$

$$C = \begin{bmatrix} c_1 \\ 0 \end{bmatrix} \tag{14}$$

$$D = \begin{bmatrix} 0 & d_1 & 0 & d_2 & 0 \\ d_m & d_n & d_l & d_c & d_e \end{bmatrix} \tag{15}$$

$$\Delta = diag\left(\frac{d}{dt}I, q_1 I, \cdots, q_n I, \Delta_1, \cdots, \Delta_n\right) \tag{16}$$

A,B,C,and D are constant matrices. q_i are arbitrary constants. Δ_i are noncommuting indeterminates. Δ is used to describe the dynamics and uncertainty of the nominal component.

Within Δ the standard dynamic element is $\frac{d}{dt}$ instead of the standard $\int$. The reason for this choice is that $\int$ is not an operator because an initial condition must be specified.

The component model of a "real" inductor is shown in Figure 13. The implicit LFT

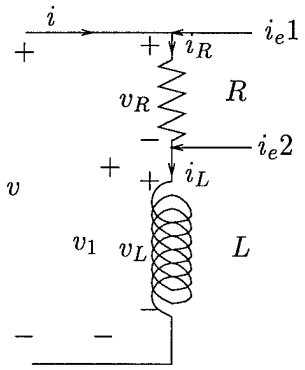

Figure 13: A Model of a "Real" Inductor

model is given by (using Figure 9,equations 13, 14, 15,and 16: $w_m = [v, i]$,$w_n = [v_L, i_L]$, $l_m = [v_R, i_R, v_1]$, $l_c = []$, $l_e = [v_{e1}, v_{e2}, i_{e1}, i_{e2}, \phi_e]$ (v_{e1} corresponds to a voltage error of R, v_{e2} corresponds to a voltage error of L, and ϕ_e corresponds to the flux error in the inductor),

$$x = \begin{bmatrix} \frac{d\phi}{dt} \\ L_0(I+\Delta_L)i_L \\ \Delta_v v_L \\ \Delta_L i_L \end{bmatrix}, A = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix},$$

$$b_1 = \begin{bmatrix} 0 & 0 \\ 0 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}, b_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

$$c_1 = \begin{bmatrix} -1 & 0 & 1 & 0 \end{bmatrix}, d_1 = \begin{bmatrix} 1 & 0 \end{bmatrix},$$

$$d_m = \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \\ 0 & 0 \end{bmatrix}, d_n = \begin{bmatrix} 0 & 0 \\ -1 & 0 \\ 0 & 0 \\ 0 & -1 \end{bmatrix}, d_l = \begin{bmatrix} -1 & 0 & -1 \\ 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 1 & 0 \end{bmatrix},$$

$$d_e = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 \end{bmatrix}, and \Delta = \begin{bmatrix} \frac{d}{dt} & 0 & 0 & 0 \\ 0 & L_0 & 0 & 0 \\ 0 & 0 & \Delta_v & 0 \\ 0 & 0 & 0 & \Delta_L \end{bmatrix}$$

The terms that are not listed are empty ie. $= []$.

2.3 Hierarchy of Components

The hierarchical structure of a system modeled using this framework is a tree (Figure 14). At each node of the tree there is a component of the form presented in section 2.2.2. Components can be ranked hierarchically if they are on the same branch of the tree. The component closer to the root is considered hierarchically above the component closer the leaf of a branch. Components not on a common branch are not related hierarchically. A particular branch may be more significant than another. For the inductor model in Figure 13, the hierarchy is

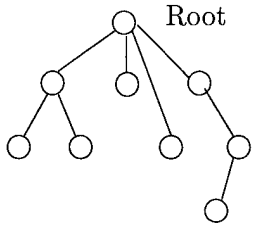

Figure 14: Hierarchical Structure

shown in Figure 15.

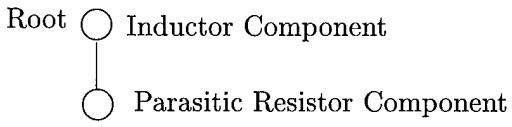

Figure 15: Hierarchical Structure of Inductor Model

The link that connects two nodes on the tree contains the information of how the latent variables of the higher component are connected to the manifest variables of the lower component. This is a hierarchical interconnection. The link defines T_1 and T_2 of (7) for the interconnection. Assuming that M_1 is the lower component and M_2 is the higher component, $T_1 = \begin{bmatrix} I & 0 & 0 & 0 & 0 \end{bmatrix}$ and $T_2 = \begin{bmatrix} 0 & 0 & t_2 & 0 & 0 \end{bmatrix}$. For the inductor model, $t_2 = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \end{bmatrix}$

2.3.1 Nonhierarchical Interconnections

Hierarchical interconnections are not the only interconnections. Any two nodes may be interconnected. The nonhierarchical interconnections are called crosstalk. Crosstalk is used to model secondary interactions between components like magnetic interconnection of disjoint inductors, gravitational interaction of masses, two disjoint flexible structures linked by air, etc. The structure of the model of the system is a web (Figure 16) rather than a tree, but the

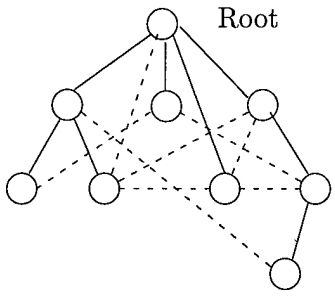

Figure 16: Web Structure of Model

web has an underlying tree structure from the hierarchy as shown in Figure 14. The solid lines in Figure 16 denote hierarchical connections and the dashed ones the crosstalk connections.

A non-hierarchical interconnection link is actually a component describing the dynamics of the connection (dynamics of air linking two disjoint flexible structures). The link defines how the latent crosstalk variables of the two hierarchical components are connected to the crosstalk component. The link defines an interconnection of 3 components. So generalizing (7) we have T_1, T_2, and T_3 where $T_i = \begin{bmatrix} 0 & 0 & 0 & t_i & 0 \end{bmatrix}$ for i=1 or 2 and $T_3 = \begin{bmatrix} t_3 & 0 & 0 & 0 & 0 \end{bmatrix}$.

2.3.2 Component Refinement

The process of component refinement is an improvement of the model of a component. A more accurate nominal model of the system (without uncertainty, noise, etc) can be parametrized. As a result, the uncertainty description of the system doesn't have to be as conservative.

Component refinement involves interconnecting a component to more "parasitic" subcomponents or describing crosstalk interaction between components, which models previously undescribed phenomenon. The proposed hierarchical modeling framework was designed with this option in mind. The idea being that a model can describe new phenomenon without having to start over. This is the purpose of the latent error variables l_e.

As a model is refined new latent manifest variables are created for components. In the circuit case these are the new interconnection variables, the voltages associated with new nodes, the error terms for new nodes and branches, and refinements of old error terms.

In the process of refining the components of a model (arriving at a more detailed model) there is an implicit reduction of the uncertainty necessary to describe the system dynamics because the uncertainty is still represented by an NCI. As more phenomenon are modeled, the inaccuracy of the model is reduced.

2.3.3 Model Reduction

Assuming that we have a detailed model of a system with all the nominal values determined, NCIs characterized, and error variables resolved, it may be desirable to view the system at dif-

ferent levels of resolution. There is a tradeoff between the complexity and fidelity of a model. When the model is used for analysis, synthesis of a controller, or simulation less detailed models may be sufficient for the task at hand which would reduce the required computation. There is no reason to use highly detailed models when crude models are sufficient.

In this framework, model reduction can be done efficiently. The model reduction process involves paring the hierarchical tree. When a branch is cut the dynamics of the removed components must be covered by uncertainty. The removed branch would be replaced by IQCs.

2.4 Hierarchical Modelling of Static Nonlinearities

When creating a mathematical model of a physical system, any degree of complexity is possible. A resistor model can simply be $e = ir$ or can include a variety of parasitic effects. When developing models for interconnection with other systems, the degree of accuracy needed is not known *a priori*. Effects critical to one system may be superfluous to another. Thus, a set of models with varying degrees of accuracy are needed. Finally, the set of models must be useful for computation.

Model hierarchies are an area of active research in the computer graphics community. Wavelets are used to create sets of models of 3 dimensional objects for on-screen rendering. As an object moves toward the foreground of an image, more complex models are used to display finer detail.[32]

Modeling for control has its own set of requirements. Primarily, simulation and analysis computations must grow reasonable as state dimension increases. In computer graphics, all problems are in 2 or 3 dimensions. Secondly, some measure of model accuracy is needed. Is a nonlinear 2 state model more accurate than a linear 5 state model? Piecewise linear (PL) systems satisfy both of these requirements.

A PL system, described in more detail in [15] and [34], is a nonlinear system of the form

$$x[k+1] = A_i x[k] + \bar{A}_i + B_i n[k],\ x \in R_i,\ i \in 1 \dots l. \tag{17}$$

where R_i is one of a finite number of regions in state space and $|n[k]| \leq 1$. This type of system is a conceptually simple extension of a linear system; each region of state space behaves as an affine system. PL systems are also easily simulated and implemented on digital computers, and can approximate a nonlinear system to any degree of accuracy.

A partial ordering of model accuracy is determined by the number of piecewise linear regions, l, and the amount of noise in the model. Larger l and less noise lead to more accurate models. There are several methods for measuring the amount of noise in the system. The simplest is the $\max_{i \in 1 \dots l} |B_i|$. More complex measures can weight the B_i or use other norms.

To demonstrate the ideas from the previous section, a model library is created for a simple static nonlinearity. The saturation

$$y = \frac{2}{\pi} \arctan u, \tag{18}$$

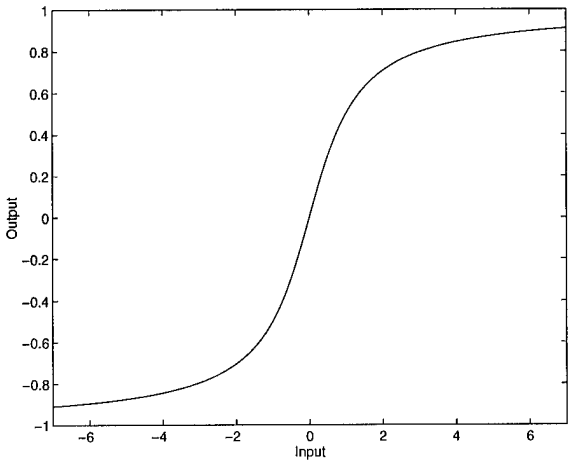

Figure 17: The saturation nonlinearity

shown in figure 17, can be described by a variety of PL systems. The simplest model is to treat it as one region, $l = 1$, with noise,

$$y = 0u + n, |n| \leq 1 \tag{19}$$

While (19) does not describe the behavior of the nonlinearity, it does describe its bounds. This model is useful when the saturation has little effect on the overall system.

A more common approximation for saturations is a 3-segment representation. When approximating (18) by a PL system with $l = 3$, some assumptions must be made. First, this approximation will hold for all values u. Second, the model will be symmetric about $u = 0$. Finally, the maximum size of the noise signal will be minimized. The model

$$y = \begin{cases} -0.87 + 0.13n & u < -2.32 \\ 0.38u + 0.13n & -2.32 \leq u \leq 2.32 \\ 0.87 + 0.13n & u > 2.32 \end{cases}, \tag{20}$$

shown in figure 18, satisfies these requirements. Note that the saturated regions, $|u| > 2.32$ are not nominally equal to the saturated values. For this model, they are offset to reduce the amount of noise needed to cover the true saturation (18). As shown by the dotted lines, there is always a noise that makes the approximate system equal the true system.

A more accurate model can be created by using a 5-segment approximation. Using the same assumptions as before, the model, shown in figure 19, is

$$y = \begin{cases} -0.95 + 0.05n & u < -6.14 \\ 0.06u - 0.55 + 0.05n & -6.14 \leq u < -1.29 \\ 0.49u + 0.05n & -1.29 \leq u \leq 1.29 \\ 0.06u + 0.55 + 0.05n & 1.29 < u \leq 6.14 \\ 0.95 + 0.05n & u > 6.14 \end{cases} \tag{21}$$

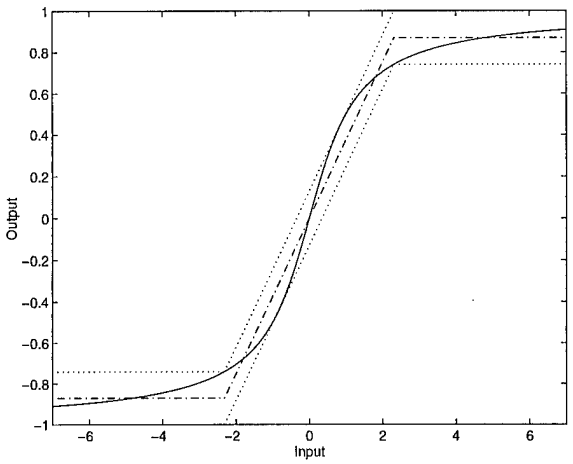

Figure 18: 3-segment saturation approximation

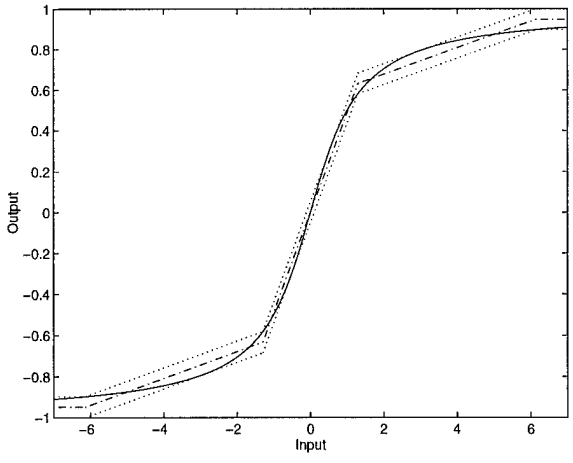

Figure 19: 5-segment saturation approximation

All three of these models are valid representations of original nonlinearity, 18. The more complex models contain less uncertainty (noise), but may require more computation during simulation.

2.4.1 Multiresolution simulation

Traditional simulation techniques yield one result for an initial condition and noise signal. If several models exist for a system, and each yields a different simulation result, which one is correct? The simulation technique should give compatible results. Also, the ability to change models during simulation is desired. One model may be accurate for one operating region and a second may describe a different region in detail.

When simulating uncertain systems, the result should be a set of all possible final conditions, not a single final condition. Since the system is uncertain, it is impossible to know the final condition exactly. The simulation should also allow a set of initial conditions, so the results of a prior simulation can be used start future runs. More accurate (less uncertain) models should yield smaller sets of final conditions than coarser models for the same set of initial conditions. Finally, all models should be compatible. For any set of initial conditions, the simulation results from all models should have some points in common. Robust simulation, described in [15] and [12], meets these requirements.

Robust simulation is the simulation of sets. For a given initial condition set and uncertainty description, the set of all possible final conditions are calculated. For model libraries that share the same state variables (or have invertible mappings between different state descriptions), any model can be used at any time step without recalculating prior results. This allows the simulation technique to choose the most appropriate model for a given initial condition set, i.e. multiresolution simulation.

The ability to do dynamic model selection is a direct result of using robust simulation. The result at any time step can be used as the input for simulating any model one time step forward. The technique does not provide guidelines for model selection; it only gives the ability to change models. As long as all models correctly describe the same system, the results are guaranteed, by construction of the algorithm, to be compatible.

This simulation technique exhibits polynomial computational growth in all variables. As problem size grows, computational cost grows reasonably. For each time step of simulation, roughly nl^2 linear programs must be solved. This technique is only applicable for models in the form of (17) and is conservative. If the results are too conservative, the simulation can be systematically refined until the exact solution is obtained.

Three one-dimensional systems are used to demonstrate the multiresolution simulation ideas. While more complex examples can easily be solved, these simple show the possible results. Robust simulation, the algorithm used during simulation, has already been successfully tested on higher order models.[15]

The first example demonstrates that in some circumstances, even the coarsest model may give acceptable results. Consider the stable system

$$x[k+1] = 0.4x[k] + 0.05\frac{2}{\pi}\arctan u[k] \tag{22}$$

time step	lower bound	exact	upper bound
0	100	100	100
1	39.95	39.95	40.05
2	15.93	15.93	16.07
3	6.32	6.32	6.48
4	2.47	2.48	2.64
5	0.94	0.96	1.11

Table 1: Simulation results for coarse model

Saturation Model	$\min x[50]$	$\max x[50]$
Model (19)	-280	2200
Model (20)	-0.47	140
Model (21)	-0.13	0.13
Exact	6.0 e-11	6.5 e-8

Table 2: Simulation results for the unstable system

with the feedback law $u[k] = -x[k]$. This system is open loop stable, and the input has very little control authority. All simulations are for 5 time steps starting from $x[0] = 100$.

The first simulation of (22), whose results are shown in table 1, uses (19) to approximate to approximate the saturation.

For this system, even the simplest model gives reasonable results. The final result, $0.942 \leq x[5] \leq 1.107$, differs by only 5% from its center value and, as expected, contains the exact result of 0.956. Note that for the first few time steps, the lower bound is, after rounding, identical to the exact value . This is because the worst case uncertainty (noise), which achieves the lower bound, is also the value needed to make the approximation match the true model.

Running the same simulation using the approximation (20) yields a final result of $0.942 \leq x[5] \leq 0.963$. This result is much tighter because the saturation only takes on values between 1 and 0.74 in the saturation region. Finally, using (21) to approximate the saturation gives $0.953 \leq x[5] \leq 0.961$. As expected, the model with the least uncertainty gives the tightest bounds.

The second example demonstrates that radically different results can be obtained from different, seemingly accurate, models. The nominally unstable system

$$x[k+1] = 1.1x[k] + \frac{2}{\pi}\arctan u[k] \tag{23}$$

is stabilized about the origin by the feedback law $u[k] = -x[k]$. What values can $x[50]$ attain if $7.5 \leq x[0] \leq 8.5$?

Table 2 shows results from the robust simulation using each of the three saturation approximations. Since the the coarsest approximation, model (19), does not account for the sign of the saturation, that model does not result in a stable system. As shown in table 2, the simulation values differ greatly from the exact values.

For model (20), which appears to approximate the nonlinearity fairly well, some states approach the origin and other states diverge. Model (21) shows that all states in the range $7.5 \leq x[0] \leq 8.5$ converge to the origin. For this example, the 3 segment saturation approximation is not sufficient; the 5 segment model is needed.

The results for models (20) and (21) indicate a limitation of the modeling framework. Since uncertainty can enter as a constant offset, the simulation never converges to an equilibrium point, but always to a ball around an equilibrium point. For example, For model (20), this set is $-0.47 < x < 0.47$. Even if $x[0] = 0$, the simulation result would be $|x[50]| < 0.47$. The size of this set is a function of the dynamics around the fixed point and the size of the model uncertainty. Model (21), which has much smaller uncertainty, reaches the ball $|x[50]| < 0.13$.

A third example demonstrates multiresolution simulation. Consider the nominally unstable system with a quantized input

$$\begin{aligned} x[k+1] &= 1.9x[k] + q(u[k]) \\ q(u) &= u - (u + 0.05) \bmod 0.1 + 0.05. \end{aligned} \tag{24}$$

While q can be exactly represented as a PL mapping, it has 10 PL regions for every unit of state space modeled. A PL mapping valid for $|x| < 10$ requires 200 PL regions. A much simpler model for the quantizer is

$$q(x) = x + 0.05n \tag{25}$$

Closing the loop with unity feedback and using approximation (25) for the quantizer gives

$$x[k+1] = 0.9x[k] + 0.05n[k]$$

Simulating the initial condition set $|x[0]| \leq 100$ for 1000 time steps gives the result $|x[1000]| \leq 0.50$, the smallest range attainable for this amount of uncertainty.

A substantially better result can be obtained by changing the quantizer approximation midway through the simulation. Another piecewise linear model for the quantizer is

$$q(x) = \begin{cases} x + 0.05n, & x \geq .55 \\ 0.5, & 0.45x \leq x < 0.55 \\ 0.4, & 0.35x \leq x < 0.45 \\ 0.3, & 0.25x \leq x < 0.35 \\ 0.2, & 0.15x \leq x < 0.25 \\ 0.1, & 0.05x \leq x < 0.15 \\ 0, & -0.05x \leq x < .05 \\ 0.1, & -0.15x \leq x < -0.05 \\ 0.2, & -0.25x \leq x < -0.15 \\ 0.3, & -0.35x \leq x < -0.25 \\ 0.4, & -0.45x \leq x < -0.35 \\ 0.5, & -0.55x \leq x < -0.45 \\ x + 0.05n, & x < -0.55 \end{cases} \tag{26}$$

This model exactly represents the quantizer near the origin, and uses the same approximation as (25) for $|x| > 0.55$.

After running the first 100 steps of the simulation using approximation (25), the range $|x[100]| \leq 0.503$ is obtained, roughly the smallest region attainable when using approximation (25). Since the region $|x[100] \leq 0.503$ is valid for any model of the system, it can be used to initialize the simulation for a different model. Switching to approximation (26) for remainder of the simulation gives the result $|x[1000]| \leq 0.095$. This range is less than one fifth the size of the result when only model (25) is used.

The multiresolution simulation not only gives a better result, but also greatly reduces computational cost. While the same result could be obtained by using model (26) for the entire simulation, the computational cost is much greater. By using (25) for the first 100 steps, about 25000 fewer linear programs are solved. Since the number of linear programs solved grows as $\mathcal{O}(tnl^2)$, this savings is even larger for higher order systems.

3 Robustness Analysis

The structured singular value, μ, introduced by Doyle in [7] has proven to be a useful framework for robustness analysis. The main advantage of the paradigm is that many robustness problems can be recast as μ problems with a particular uncertainty description.

Although computing μ is NP-hard [4], computationally tractable upper and lower bounds exist [39, 26, 28, 38, 41, 2]. For some problems, the gap between the upper and lower bounds may be large. Strategies for reducing the gap are based upon branch and bound techniques [27]. The branch and bound algorithms involve dividing a μ problem into two μ problems and repeating this process until the gap is acceptable. Critical to these techniques is preventing exponential growth in the number of active μ problems. For existing branch and bound algorithms for μ, the branching is limited to axially aligned branches.

An extension of the branch and bound algorithm leads to probabilistic μ [43, 19]. Probabilistic μ seems to be computationally intractable even for low dimensional problems (≥ 4), the bounds are very conservative for any reasonable computation time. Even though the branch and bound and probabilistic μ algorithms are trivial to parallelize, there are still fundamental improvements that can be made in the algorithms.

The focus of this work is developing a richer class of uncertainty descriptions for which analogous bounds to μ can be computed. Traditionally, this has meant adding to the possible uncertainty structures, like real parameters. This work takes a different approach. Given a matrix and a block structure for the uncertainty, $\frac{1}{\mu}$ defines a distance to singularity of $(I - M\Delta)$ in the space of allowable Δ. For the standard μ problem this distance is measured using the ∞-norm. It turns out that analogous bounds can be derived if the distance is measured using the 2-norm.

With the development of the new set descriptions, it is possible to do more creative branching, which may result in better algorithms for probabilistic μ and branch and bound for μ. There may also exist physical problems which fundamentally have spherical constraints on the uncertainty, but this is not the main motivation for the results in this work.

3.1 Preliminaries

The notation is standard. If M is a matrix, then M^T, M^* denote the transpose and conjugate transpose matrices, respectively. The $n \times n$ identity matrix will be denoted by I_n, I is used when the dimension is obvious. $\mathbb{1}_n$ is an $n \times n$ matrix of 1's. $diag[A_1, \cdots, A_n]$ is a block diagonal matrix with A_i on the diagonal. The $vec(M)$ is a vector which is the columns of M stacked into a vector.

The spectral radius, real spectral radius, and maximum positive real eigenvalue of a square matrix M are denoted by $\rho(M)$, $\rho_r(M)$, and $\overline{\lambda}_r(M)$, respectively. The maximum and minimum singular values of a matrix M are denoted by $\overline{\sigma}(M)$ and $\underline{\sigma}(M)$ respectively. $Tr(M)$ is the trace of M. $det(M)$ is the determinant of M. The Frobenius norm of a matrix M is defined as $\|M\|_F = \sum_{ij} |m_{ij}|^2$. A Hermitian matrix $M^* = M \in \mathbb{C}^{n \times n}$ is said to be positive (semi) definite if $x^* M x > 0 (\geq 0)$ for all nonzero $x \in \mathbb{C}^n$.

The Kronecker product of two matrices, A and B, is denoted by $A \otimes B$. The Hadamard (or Schur) element-wise product of two matrices $A = [a_{ij}]$ and $B = [b_{ij}]$ of the same dimensions is defined as $A \circ B \equiv [a_{ij} b_{ij}]$. In this work the Hadamard product is sometimes used to specify matrix structure. For example, to specify that a matrix is diagonal the notation $I \circ D = D$ is used. Some useful properties of the Hadamard product follow.

Lemma 1 *If $A = I \circ A$ and $B = I \circ B$ then $AXB = X \circ (a * b^T)$ where $A = diag[a_1, \cdots, a_n]$ and $B = diag[b_1, \cdots, b_n]$.*

Lemma 2 (Schur Product Theorem) *[11] If $X_i > (\geq) 0$ and $X = X_1 \circ X_2$ then $X > (\geq) 0$*

The fundamental picture in the μ-paradigm [28] is shown in Figure 20. Where M is a matrix and Δ is a matrix in a set of allowable matrices. The set of allowable Δ is described by block diagonal structure, and a size constraint, $\overline{\sigma}(M) < 1$. The fundamental question in

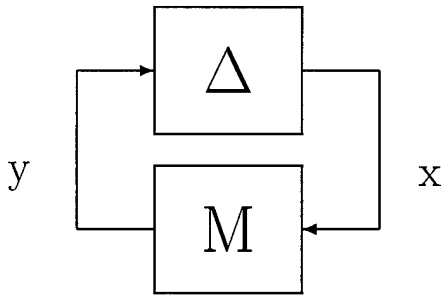

Figure 20: Standard Interconnected System

the μ-paradigm is are there any nontrivial solutions to the loop equations shown in Figure 20. $\mu(M)$ answers that question.

Definition 3 *The uncertainty structure,*

$$\mathbf{\Delta} := \{diag\left[\delta_{r1}I_{k_{r1}}, \cdots, \delta_{rn_R}I_{k_{rn_R}}, \delta_{c1}I_{k_{c1}}, \cdots, \delta_{cn_C}I_{k_{cn_C}}, \Delta_{f1}, \cdots, \Delta_{fn_F}\right] : \\ \delta_{ri} \in \mathbb{R}, \delta_{ci} \in \mathbb{C}, and \Delta_{fi} \in \mathbb{C}^{n_{fi} \times n_{fi}}\}. \tag{27}$$

Definition 4 *[42] For $M \in \mathbb{R}^{n \times n}$, $\mu_{\mathbf{\Delta}}$ is defined as*

$$\mu_{\mathbf{\Delta}}(M) := \frac{1}{\min\{\overline{\sigma}(\Delta) : \Delta \in \mathbf{\Delta}, \det(I - M\Delta) = 0\}} \tag{28}$$

unless no $\Delta \in \mathbf{\Delta}$ makes $I - M\Delta$ singular, in which case $\mu_{\mathbf{\Delta}}(M) := 0$.

Alternatively equation 28 can be written as

$$\mu_{\mathbf{\Delta}}(M) = \sup_{\Delta \in \mathbf{\Delta}, \overline{\sigma}(\Delta) < 1} \rho_r(M\Delta) = \sup_{\Delta \in \mathbf{\Delta}, \overline{\sigma}(\Delta) < 1} \overline{\lambda}_r(M\Delta) \tag{29}$$

With the appropriate projection the uncertainty set in equation 29 defines a hypercube of dimension $n_R + n_C + n_F$.

Computing $\mu_{\mathbf{\Delta}}(M)$ is NP-hard but computationally tractable upper and lower bounds exist. The lower bound computation is based upon equation 29. Choose any $\Delta \in \mathbf{\Delta}$, then $\rho_r(M\Delta)$ is a lower bound to $\mu_{\mathbf{\Delta}}(M)$. The lower bound algorithms involve strategies to choose Δ to maximize $\overline{\lambda}_r(M\Delta)$ and is a non-convex optimization problem. The upper bound which is a Linear Matrix Inequality (LMI),

$$\mu_{\mathbf{\Delta}}(M) \leq \overline{\mu}_{\mathbf{\Delta}}(M) = \inf_{D \in \mathbf{D}, G \in \mathbf{G}} \{\gamma : M^*DM + j(GM - M^*G) - \gamma^2 D < 0\} \tag{30}$$

is a convex optimization problem where,

$$\mathbf{D} = \{Diag\left[D_{r1}, \cdots, D_{rn_R}, D_{c1}, \cdots, D_{cn_C}, d_1 I_{n_f 1}, \cdots, d_{fn_F} I_{n_f F}\right] > 0\} \text{ and} \tag{31}$$

$$\mathbf{G} = \{Diag\left[G_{r1}, \cdots, G_{rn_R}, 0, \cdots, 0\right] : G_{ri} = G_{ri}^*\} \tag{32}$$

To define μ, a matrix and a uncertainty structure, $\mathbf{\Delta}$, must be specified. In this presentation, notation for standard μ is $\mu_{\mathbf{\Delta}}(M)$. For uncertainty with spherical and elliptical constraints, $\mu_{s,\mathbf{\Delta}}(M)$ and $\mu_{e,\mathbf{\Delta}}(M)$ are used. For a μ problem with combinations of spherical, elliptical, and standard constraints, $\mu_{g,\mathbf{\Delta}}(M)$ is used. If a μ problem is specified without an uncertainty structure, the default uncertainty structure is indicated. The default uncertainty structure is specified in definition 5.

3.2 Spherical μ

For simplicity in derivation and presentation it is assumed that the uncertainty consists of only complex scalars, ie no full blocks, repeated scalars, or real parameters, and the uncertainty set is described by a Frobenius norm constraint rather than an induced norm constraint like $\overline{\sigma}(\Delta) \leq 1$.

Definition 5 *The default uncertainty structure is*

$$\mathbf{\Delta_s} := \{diag\,[\delta_1, \cdots, \delta_n] : \delta_i \in \mathbb{C}\}. \tag{33}$$

The level curves of $\|\Delta\|_F$ are defined by

$$\sum_{i=1}^{n} |\delta_i|^2 = K, \tag{34}$$

which describes hyperspheres in $\mathbf{\Delta_s}$ space, hence the name spherical μ (μ_s).

Definition 6 *For $M \in \mathbb{R}^{n\times n}$, Spherical μ, μ_s is defined as*

$$\mu_s(M) := \frac{1}{\min\{\|\Delta\|_F : \Delta \in \mathbf{\Delta_s}, \det(I - M\Delta) = 0\}} \tag{35}$$

unless no $\Delta \in \mathbf{\Delta_s}$ makes $I - M\Delta$ singular, in which case $\mu_s(M) := 0$.

Alternatively equation 35 can be written as

$$\mu_s(M) = \sup_{\Delta\in\mathbf{\Delta_s}, \|\Delta\|_F<1} \rho_r(M\Delta) \tag{36}$$

Much like the standard μ problem, computing spherical μ exactly seems to be intractable. As a result, one is resigned to computable upper and lower bounds as in the standard μ case.

Theorem 7 $\mu_s(M) \leq \mu(M) \leq \sqrt{n}\mu_s(M)$

Proof. The statement follows immediately from

$$\{\Delta : \Delta \in \mathbf{\Delta_s}, \|\Delta\|_F < 1\} \subseteq \{\Delta : \Delta \in \mathbf{\Delta_s}, \overline{\sigma}(\Delta) < 1\} \tag{37}$$

$$\{\Delta : \Delta \in \mathbf{\Delta_s}, \overline{\sigma}(\Delta) < 1\} \subseteq \{\Delta : \Delta \in \mathbf{\Delta_s}, \|\Delta\|_F < \sqrt{n}\} \tag{38}$$

and equations 36 and 29.

◇

As a consequence of Theorem 7 any upper bound of $\mu(M)$ is also an upper bound $\mu_s(M)$ but this would be a conservative bound which scales badly with dimension.

3.2.1 An Upper bound to Spherical μ

A $\overline{\mu}_s(M)$, which is not necessarily an upper bound of $\mu(M)$, can be computed by solving an LMI similar to equation 30. The construction of the LMI follows.

Lemma 8 (Schur Complement) *[3] The matrix inequality,*

$$\begin{bmatrix} Q & S \\ S^* & R \end{bmatrix} > 0 \tag{39}$$

with $Q = Q^*$ and $R = R^*$ is equivalent to

$$Q - SR^{-1}S^* > 0 \tag{40}$$

$$R > 0 \tag{41}$$

The following Lemma is used in a separating hyperplane argument to construct a sufficient LMI condition for there being no nontrivial solutions to a quadratic form.

Lemma 9 *If* $A > 0$ *and* $B > (\geq)0$ *then* $Tr(AB) > (\geq)0$.

Proof.

$$Tr(AB) = Tr(AB^{\frac{1}{2}}B^{\frac{1}{2}}) = Tr(B^{\frac{1}{2}}AB^{\frac{1}{2}}) \tag{42}$$

$$B^{\frac{1}{2}} = (B^{\frac{1}{2}})^* \Rightarrow B^{\frac{1}{2}}AB^{\frac{1}{2}} > (\geq)0 \Rightarrow Tr(B^{\frac{1}{2}}AB^{\frac{1}{2}}) > (\geq)0 \tag{43}$$

◇

Theorem 10

$$\mu_s(M) \leq \overline{\mu}_s(M) = \inf_{D>0}\{\gamma : M^*(I \circ D)M - \gamma^2 D < 0\} \tag{44}$$

Proof. x and y are the output and input vectors of Δ, respectively.

$$\sum_{i=1}^{n} |\delta_i|^2 < \gamma^{-2} \tag{45}$$

$$\Leftrightarrow \delta^*\delta < \gamma^{-2} \text{ for } \delta = [\delta_1 \ \delta_2 \cdots \delta_n]^T \tag{46}$$

$$\Leftrightarrow \begin{bmatrix} \gamma^{-2} & \delta^* \\ \delta & I \end{bmatrix} > 0 \text{ Applying Lemma 8} \tag{47}$$

Using $x_i = \delta_i y_i$,

$$\Rightarrow \begin{bmatrix} \gamma^{-2} & x^* \\ x & I \circ (yy^*) \end{bmatrix} \geq 0 \text{ Applying Lemma 1} \tag{48}$$

$$\Leftrightarrow I \circ (yy^*) - \gamma^2 xx^* \geq 0 \text{ Applying Lemma 8} \tag{49}$$

Using $y = Mx$,

$$\mu_s(M) < \gamma \Leftrightarrow y = x = 0 \text{ is the only solution to equation 49} \tag{50}$$

Using Lemma 9,

$$\begin{gathered} \text{If } \exists D > 0 : Tr(D(I \circ ((Mx)(Mx)^*) - \gamma^2 xx^*)) < 0, \forall x \in \mathbb{C}^n \\ \text{then equation 49 has no nontrivial solutions} \end{gathered} \tag{51}$$

$$Tr(D(I \circ (yy^*) - \gamma^2 xx^*)) < 0, \forall x \in \mathbb{C}^n \tag{52}$$

$$\Leftrightarrow x^*M^*(I \circ D)Mx - \gamma^2 x^* Dx < 0, \forall x \in \mathbb{C}^n \tag{53}$$

$$\Leftrightarrow M^*(I \circ D)M - \gamma^2 D < 0 \tag{54}$$

◇

The main difference between this derivation and the $\overline{\mu}(M)$ formulation is in deriving the quadratic form in equation 49. The rest of the derivation corresponds to the standard separating plane or S-procedure argument [23].

Another upper bound to $\mu_s(M)$ can be computed by using the implicit μ-framework presented in Newlin [25] and Tierno [36]. From initial investigation, the implicit bound is more conservative than the bound presented here.

3.2.2 Properties of $\overline{\mu}_s$

The LMI optimization in Theorem 10 can be solved by using general purpose LMI solvers, but it can be solved more efficiently,

$$\inf_{D>0}\{\gamma: M^*(I\circ D)M-\gamma^2 D<0\}=\sqrt{\rho(M^T\circ M^*)}. \tag{55}$$

The reader is refer ed to [30] for more details.

The LMI in equation 44 is related to the standard $\overline{\mu}(M)$ LMI, but it is unclear whether the new LMI is consistent with the standard LMI. Corollary 11 and Theorem 13 are presented to validate that the two bounds are consistent.

Corollary 11 $\overline{\mu}_s(M)=\overline{\mu}_s(TMT^{-1})$ *where* $I\circ T=T>0$

Proof. The LMI in equation 54 becomes

$$T^{-*}M^*T^*(I\circ D)TMT^{-1}-\gamma^2 D<0 \tag{56}$$

$$M^*T^*(I\circ D)TM-\gamma^2 T^*DT<0 \tag{57}$$

Let $T=diag[t_1,\cdots,t_n]$:

$$M^*(I\circ D\circ(t^*t))M-\gamma^2(D\circ(t^*t))<0 \text{ Applying lemma 1} \tag{58}$$

Let $\overline{D}=D\circ(t^*t)$:

$$M^*(I\circ\overline{D})M-\gamma^2\overline{D}<0 \tag{59}$$

$\overline{D}>0$ which follows from lemma 2 $\Rightarrow$ for a given γ, if the LMI in equation 56 has a solution then the LMI in equation 54 has a solution.

$$\Rightarrow \overline{\mu}_s(M)\le\overline{\mu}_s(TMT^{-1})\le\overline{\mu}_s(T^{-1}TMT^{-1}T)=\overline{\mu}_s(M) \tag{60}$$

$\diamond$

In Corollary 11, T enters in exactly like the D-scales from the standard μ upper bound. This implies that $\overline{\mu}_s(M)$ has exploited all the structure in the uncertainty set that $\overline{\mu}(M)$ has and that the additional freedom in the $\overline{\mu}_s(M)$ LMI is related to the structure of the Frobenius set description. In the standard case, the hypercube does not provide additional structure to exploit.

The following lemma is a useful tool in relating $\overline{\mu}_s(M)$ and $\overline{\mu}(M)$.

Lemma 12 *$D \leq nI \circ D$ for $D > 0$*

Proof.

$$\overline{\sigma}(\mathbb{1}_n) = n \text{ and } \underline{\sigma}(nI) = n \tag{61}$$

$$\Rightarrow nI - \mathbb{1}_n \geq 0 \tag{62}$$

$$(nI \circ D) - D = (nI - \mathbb{1}_n) \circ D \tag{63}$$

Applying Lemma 2,

$$(nI \circ D) - D \geq 0 \tag{64}$$

$\diamond$

Theorem 13 *$\overline{\mu}_s(M) \leq \overline{\mu}(M) \leq \sqrt{n}\overline{\mu}_s(M)$*

Proof. Equations 30 and 54 can be rewritten as

$$\overline{\mu}(M) = \inf_{D_1 > 0} \{\gamma : M^*(I \circ D_1)M < \gamma^2 (I \circ D_1)\} \tag{65}$$

and

$$\overline{\mu}_s(M) = \inf_{D_2 > 0} \{\gamma : M^*(I \circ D_2)M < \gamma^2 D_2\} \tag{66}$$

respectively. Assume $D_2 = I \circ D_1$

$$\Rightarrow \overline{\mu}_s(M) \leq \overline{\mu}(M) \tag{67}$$

Apply Lemma 12:

$$M^*(I \circ D_2)M < (\overline{\mu}_s(M))^2 D_2 \leq n(\overline{\mu}_s(M))^2 (I \circ D_2) \tag{68}$$

Assume $D_1 = D_2$:

$$\Rightarrow \overline{\mu}(M) \leq \sqrt{n}\overline{\mu}_s(M) \tag{69}$$

$\diamond$

Theorems 7 and 13 indicate further consistency between the spherical μ upper bound, the standard μ upper bound, and the set containment shown in figure 21. If the relationship between $\mu_s(M)$ and $\mu(M)$ was not the same as the relationship between their respective upper bounds, then their would be instances where one of the upper bounds could be trivially improved. For example, assume that the relationship from Theorem 13 was replaced by $\overline{\mu}_s(M) \leq \overline{\mu}(M) \leq n\overline{\mu}_s(M)$. If both of these upper bounds were computed with $\overline{\mu}_s(M) = 1$ and $\overline{\mu}(M) = n$. Then latter upper bound would be useless, because the former implies that $\mu_s(M) \leq 1$. This in turn implies that $\mu(M) \leq \sqrt{n}$ from Theorem 7. So $\overline{\mu}(M)$ could be improved.

$\mu(M)$ is equal to the upper bound in the rank one case. The exactness of the upper bound for the rank one case is a direct consequence of convexity. In fact, in this case the determinant is linear in the uncertain parameters. The same properties hold for $\mu_s(M)$ in the rank one case.

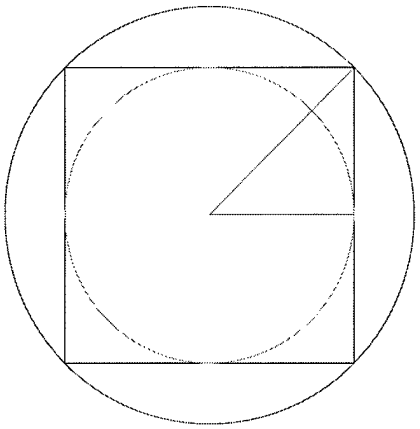

Figure 21: Cube, Inscribed, and Superscribed Spheres

Theorem 14 $\overline{\mu}_s(M) = \mu_s(M)$ *for* $rank(M) = 1$

Proof. Consider the rank one matrix M. Applying the explicit solution given in equation 55, the optimal value of γ is $\gamma = \sqrt{\sum_{i=1}^{n} |m_{ii}|^2}$. An explicit perturbation that makes $(I - M\Delta)$ singular is $\Delta = (I \circ M')/\gamma^2$. Since the Frobenius norm of Δ is $1/\gamma$, then γ is also a lower bound for μ_s. Therefore, $\mu_s = \overline{\mu_s}$ in the rank one case.
◇

The worst ratio between $\mu_s(M)$ and $\overline{\mu}_s(M)$ is at least $\sqrt{n}$. This ratio is achieved by

$$M = \begin{bmatrix} 0 & 1 \\ I & 0 \end{bmatrix}, \tag{70}$$

for which $\mu_s(M) = \frac{1}{\sqrt{n}}$ and $\overline{\mu}_s(M) = 1$.

3.2.3 Generalizations of μ_s

The solution for the upper bound of spherical μ can be generalized from an uncertainty described by a spheres to hyperellipsoids. The upper bound can also be generalized to other traditional uncertainty descriptions like repeated scalars, real parameters, and full blocks. These generalizations can be mixed and matched to form a generalization of the LMI in equation 30. Only the resulting upper bound LMIs will be presented. It is relatively straight forward to extend the derivation of the LMI in Theorem 10 to the results presented.

A natural generalization of the hypersphere is the hyperellipsoids,

$$\delta^* P \delta < 1, P > 0. \tag{71}$$

μ for a hyperellipsoid is defined by

$$\mu_e(M) := \frac{1}{\min\{\delta^* P \delta : \Delta \in \mathbf{\Delta_s}, \det(I - M\Delta) = 0\}} \tag{72}$$

It is trivial to generalize $\overline{\mu}_s$, equation 44, when P is a diagonal matrix to compute an upper bound,$\overline{\mu}_e(M)$, to elliptical μ, $\mu_e(M)$. For this case, the principle axes of the ellipsoid are aligned with the coordinate axes in $\mathbf{\Delta_s}$. The matrix M can be rescaled to create a new and equivalent problem where the uncertainty can be described by a spherical constraint. This rescaling is analogous to scaling a rectangle into a square. The LMI in equation 44 can be used to compute an upper bound to the original elliptical μ problem.

For the general hyperellipsoid in equation 71, the proof in Theorem 10 can be generalized to construct an upper bound LMI solution. The LMI for $\overline{\mu}$ on the general hyperellipsoid is given by

$$\mu_e(M) \leq \overline{\mu}_e(M) = \inf_{D>0}\{\gamma: M^*(P^{-1} \circ D)M < \gamma^2 D\}. \tag{73}$$

For $\overline{\mu}_s(M)$, $P = P^{-1} = I$ and equation 73 reduces to equation 66. For the LMI in equation 73, a closed form expression for the optimal gamma exists,

$$\overline{\mu}_e(M) = \sqrt{\rho((M^T \otimes M^*)diag(vec(P^{-1})))}, \tag{74}$$

the reader is referred to [30].

Within the μ framework an allowable uncertainty structure is a repeated scalar [],

$$\mathbf{\Delta_{ms}} := \{diag\,[\delta_1 I_{c_1}, \cdots, \delta_n I_{c_n}] : \delta_i \in \mathbb{C}\}. \tag{75}$$

This uncertainty structure can be fit into a spherical context. For notational convenience, a matrix

$$Q := diag(\mathbb{1}_{c_1}, \cdots, \mathbb{1}_{c_n}), \tag{76}$$

is introduced. The resulting upper bound LMI is given by

$$\mu_{s,\mathbf{\Delta_{ms}}}(M) \leq \overline{\mu}_{s,\mathbf{\Delta_{ms}}}(M) = \inf_{D>0}\{\gamma: M^*(Q \circ D)M < \gamma^2 D\}. \tag{77}$$

In equation 77, Q is not an invertible matrix. Q enters the LMI as a term of a Hadamard product, as a result the invertibility of Q is irrelevant. The closed form solution is identical to equation 74 with P^{-1} replaced by Q.

The uncertainty structure for real parameters is described by,

$$\mathbf{\Delta_{rs}} := \{diag\,[\delta_1, \cdots, \delta_n] : \delta_i \in \mathbb{R}\}. \tag{78}$$

The additional freedom in the LMI in equation 30 of the G-scales corresponds to the constraint that δ_i is real and that the input and output signals of the block must be aligned in $\mathbb{C}$. The G-scales are independent of the magnitude and linkage between separate blocks. The resulting upper bound LMI is

$$\begin{gathered}\mu_{s,\mathbf{\Delta_{rs}}}(M) \leq \overline{\mu}_{s,\mathbf{\Delta_{rs}}}(M) = \\ \inf_{D>0,G=G^*}\{\gamma: M^*(I \circ D)M + j((I \circ G)M - M^*(I \circ G)) < \gamma^2 D\}.\end{gathered} \tag{79}$$

The uncertainty structure for only full block uncertainty is

$$\mathbf{\Delta_{fs}} := \{diag\,[\Delta_{f1}, \cdots, \Delta_{fn_F}] : \Delta_{fi} \in \mathbb{C}^{n_{fi} \times n_{fi}}\}. \tag{80}$$

Computing an upper bound for mu with spherical constraints $\overline{\sigma}(\Delta_i)$ which includes non-scalar Δ_i,

$$\sum_{i=1}^{n} |\overline{\sigma}(\Delta_i)|^2 < 1, \tag{81}$$

is not a natural extension of the LMI for diagonal uncertainty. For diagonal uncertainty in the standard μ problem the structure of the D-scales can be specified by $Q \circ D$ where D is full. For full blocks the resulting structure in the D-scales can not be specified using the Hadamard product of a constant matrix and a full D.

One method to construct an upper bound is to convert the original spherical $\mu_{s,\mathbf{\Delta_{fs}}}(M)$ problem with full blocks into a new problem, $\mu_s(\overline{M})$ where $\mu_{s,\mathbf{\Delta_{fs}}}(M) \leq \mu_s(\overline{M})$. $\overline{M}$ is constructed from M. M must be partitioned into M_{ij} which is compatible with the block structure of the uncertainty. What is meant by this is that M_{ij} maps the output of the j^{th} block of Δ to the input of the i^{th} block of Δ. $\overline{M}$ is constructed as follows,

$$\overline{M}_{ij} = \overline{\sigma}(M_{ij}) \tag{82}$$

The new problem has a smaller dimension matrix than the original problem and the upper bound bound has a closed form expression, but this bound will tend to be conservative because $\mu_{s,\mathbf{\Delta_{fs}}}(M) \leq \mu_s(\overline{M})$.

Another method is to construct a larger problem, larger dimension and more blocks in Δ, for which $\mu_{s,\mathbf{\Delta_{fs}}}(M) = \sqrt{\mu_{g,\mathbf{\Delta_{new}}}(\overline{M})}$ and $\overline{\mu}_{g,\mathbf{\Delta_{new}}}(\overline{M})$ can be computed. The original

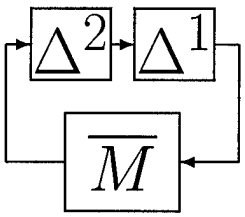

Figure 22: Expanded System

system shown in figure 20 has a spherical constraint on Δ. This is equivalent to the problem shown in figure 22, where

$$\mathbf{\Delta^1} = diag(\delta_1 I_{N_1}, \cdots, \delta_n I_{N_n}), \delta_i \in \mathbb{C} \tag{83}$$

with the constraint that

$$|\delta_i| \leq 1, \forall i \tag{84}$$

and

$$\mathbf{\Delta^2} = \mathbf{\Delta_{fs}}. \tag{85}$$

The spherical constraint can be moved from Δ^2 to Δ^1 to form an equivalent system. So equation 84 is replaced by

$$\sum_{i=1}^{n} |\delta_i|^2 \leq 1 \tag{86}$$

and the structure of Δ is unchanged but equation 81 is replaced by

$$\overline{\sigma}(\Delta_i^2) \leq 1 \forall i. \tag{87}$$

The system shown in figure 22 can be put into the original form as shown in figure 20. Let

$$\overline{\mathbf{\Delta}} = \left\{ \begin{bmatrix} \Delta^1 & 0 \\ 0 & \Delta^2 \end{bmatrix} : \Delta^1 \in \mathbf{\Delta^1}, \Delta^2 \in \mathbf{\Delta^2} \right\} \tag{88}$$

$$\overline{M} = \begin{bmatrix} 0 & I \\ M & 0 \end{bmatrix} \tag{89}$$

By construction,

$$\sqrt{\mu_{g,\overline{\mathbf{\Delta}}}(\overline{M})} = \mu_{s,\mathbf{\Delta_{fs}}}(M). \tag{90}$$

An upper bound of $\mu_{g,\overline{\mathbf{\Delta}}}(\overline{M})$ is given by

$$\inf_{D_1>0, D_2 \in \mathbf{D_2}} \left\{ \gamma : \begin{array}{c} Q \circ D_1 < \gamma^2 D_2 \\ M^* D_2 M < \gamma^2 D_1 \end{array} \right\} \tag{91}$$

where Q is similar to equation 76, and $\mathbf{D_2}$ is the standard μ D-scales for full block, ie it consists of $\delta_i I_{n_i}$ blocks.

References

[1] C. Beck, R. D'Andrea, F. Paganini, W.-M. Lu, and J. Doyle. A state space theory of uncertain systems. In *Proc. Int. Fed. Auto. Control*, 1996.

[2] C. Beck and J. C. Doyle. Mixed μ upper bound computation. In *Proceedings of the 31st Conference on Decision and Control*, pages 3187–3192, 1992.

[3] S. P. Boyd, L. E. Ghaoui, E. Feron, and V. Balakrishnan. *Linear Matrix Inequalities in System and Control Theory.* SIAM, 1994.

[4] R. D. Braatz, P. M. Young, J. C. Doyle, and M. Morari. Computational complexity of μ calculation. *IEEE Transactions on Automatic Control*, 39:1000–1002, 1994.

[5] R. D'Andrea and F. Paganini. Interconnection of uncertain behavioral systems for robust control. In *Proceedings of the 32nd Conference on Decision and Control*, 1993.

[6] R. D'Andrea and F. Paganini. Why behave? In *Proceedings of the 32nd Conference on Decision and Control*, 1993.

[7] J. C. Doyle. Analysis of feedback systems with structured uncertainty. *IEE Proceedings, Part D*, 129(6):242–250, Nov. 1982.

[8] C. E. García, D. M. Prett, and M. Morari. Model predictive control: Theory and practice — A survey. *Automatica*, 25(3):335–348, May 1989.

[9] G. H. Golub and C. F. V. Loan. *Matrix Computations.* The Johns Hopkins University Press, Baltimore, 1987.

[10] L. Gurvits. Stability of discrete linear inclusion. *Linear Algebra and its Applications*, 231:47–85, December 1995.

[11] R. A. Horn and C. R. Johnson. *Matrix Analysis.* Cambridge University Press, New York, 1985.

[12] M. Kantner. Robust simulation home page. Available on-line at `http:\\ www.cds.caltech.edu\~kantner\robustsim`.

[13] M. Kantner. Hierarchical modeling and multiresolution simulation. In *Proceedings of the American Control Conference*, 1997.

[14] M. Kantner. Robust stability of piecewise linear discrete time systems. In *Proceedings of the American Control Conference*, 1997.

[15] M. Kantner and J. Doyle. Robust simulation and nonlinear performance. In *Proc. IEEE Conference on Decision and Control*, December 1996.

[16] M. Kantner and J. Primbs. Nonlinear mpc lower bounds via robust simulation. In *Proceedings of the American Control Conference*, 1997.

[17] N. K. Karmarkar. A new polynomial–time algorithm for linear programming. *Combinatorica*, 4:373–395, 1984.

[18] S. Khatri, R. D'Andrea, and J. Doyle. Uncertain hierarchical modeling. In *Proc. IEEE Conference on Decision and Control*, December 1996.

[19] S. Khatri and P. A. Parrilo. Guaranteed bounds for probabilistic μ. In *Proceedings of the American Control Conference*, 1988.

[20] S. Khatri and P. A. Parrilo. Spherical μ. In *Proceedings of the American Control Conference*, 1988.

[21] D. Mayne and H. Michalska. Receding horizon control of nonlinear systems. *IEEE Trans. Aut. Control*, 35:814–824, 1990.

[22] A. Megretski and S. Treil. S-procedure and power distribution inequalities: A new method in optimization and robustness of uncertain systems. *Mittag-Leffler Institute*, 1991.

[23] A. Megretski and S. Treil. Power distribution inequalities in optimization and robustness of uncertain systems. *Journal of Mathematical Systems, Estimation, and Control*, 3, No. 3:301–319, 1993.

[24] V. Nevistić and J. Primbs. Constrained nonlinear optimal control: A converse hjb approach. CDS Technical Memo CIT-CDS 96-021, California Institute of Technology, Pasadena, CA 91125, 1996.

[25] M. Newlin. *Model Validation, Control, and Compoutation.* PhD thesis, California Institute of Technology, 1996.

[26] M. P. Newlin and S. T. Glavaski. Advances in the computation of the μ lower bound. In *Proceedings of the American Control Conference*, pages 442–446, 1995.

[27] M. P. Newlin and P. M. Young. Mixed μ problems and branch and bound techniques, 1996. Submitted to International Journal of Robust and Nonlinear Control, Special Issue on Multivariable Stability Margin.

[28] A. K. Packard and J. C. Doyle. The complex structured singular value. *Automatica*, 29:71–109, 1993.

[29] F. Paganini. *Sets and Constraints in the Analysis of Uncertain Systems.* PhD thesis, California Institute of Technology, 1995.

[30] P. A. Parrilo and S. Khatri. Closed form solutions for a class of LMIs. In *Proceedings of the American Control Conference*, 1998.

[31] N. Pettit and P. Wellstead. A graphical analysis method for piecewise linear systems. In *Proc. IEEE Conference on Decision and Control*, pages 1122–1127, December 1994.

[32] P. Schröder. Wavelets in computer graphics. *Proceedings of the IEEE*, to appear; 1996. Invited article for special issue on wavelets, Ingrid Daubechies and Jelena Kovačević, Eds.

[33] E. Sontag. Nonlinear regulation: The piecewise linear approach. *IEEE Transactions on Automatic Control*, 26(2):346–357, April 1981.

[34] E. Sontag. Remarks on a picewise-linear algebra. *Pacific Journal of Mathematics*, 98(1):183–201, 1982.

[35] E. Sontag. From linear to nonlinear: Some complexity comparisons. In *Proc. IEEE Conference on Decision and Control*, pages 2916–2920, December 1995.

[36] J. Tierno. *A Computational approach to nonlinear system analysis.* PhD thesis, California Institute of Technology, 1996.

[37] J. C. Willems. Paradigms and puzzles in the theory of dynamical systems. *IEEE Transactions on Automatic Control*, 1991.

[38] P. Young, M. Newlin, and J. Doyle. Structured singular value analysis with real and complex uncertainties. In *Proceedings of the 30^{th} Conference on Decision and Control*, 1991.

[39] P. M. Young and J. C. Doyle. A lower bound for the mixed μ problem. Submitted to IEEE Transactions on Automatic Control.

[40] P. M. Young, M. P. Newlin, and J. C. Doyle. Let's get real. CDS Technical Memo CIT-CDS 92-001, California Institute of Technology, Pasadena, CA 91125, Sept. 1992.

[41] P. M. Young, M. P. Newlin, and J. C. Doyle. Let's get real. In *Robust Control Theory*, pages 143–173. Springer-Verlag, 1995. IMA Proceedings Volume 66.

[42] K. Zhou, K. Glover, and J. Doyle. *Robust and Optimal Control.* Prentice Hall, 1995.

[43] X. Zhu, Y. Huang, and J. Doyle. Soft vs. hard bounds in probablistic robustness analysis. In *Proceedings of the American Control Conference*, 1996.